BAKALÁŘSKÁ PRÁCE

Učební text pro výuku odborného předmětu

Textbook for technical subject teaching

STUDIJNÍ PROGRAM

Specializace v pedagogice

STUDIJNÍ OBOR

Učitelství odborných předmětů

VEDOUCÍ PRÁCE

† prof. RNDr. Emanuel Svoboda, CSc.

PhDr. Blanka Jirkovská, Ph.D.

KLIMEŠ

ONDŘEJ TOMÁŠ

2020

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: Klimeš Jméno: Ondřej Tomáš Osobní číslo: klimeon6

Fakulta/ústav: Masarykův ústav vyšších studií (MÚVS)

Zadávající katedra/ústav: Oddělení pedagogických a psychologických studií

Studijní program: Specializace v pedagogice (B 7507)

Studijní obor: Učitelství odborných předmětů (7504R100)

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Učební text pro výuku odborného předmětu

Název bakalářské práce anglicky:

Workbook for teaching specialized subject

Pokyny pro vypracování:

CÍL: Návrh témat a tvorba pracovních listů pro praktickou výuku programovacího jazyka C# pro studenty 3. ročníku zaměření 18–20–M/01 - INFORMAČNÍ TECHNOLOGIE

PŘÍNOS: Výuka programování má vést nejen k získání znalostí, ale také k rozvoji dovedností řešit nové úkoly, osvojení vývojových prostředí, získání zkušenosti s časovou a systémovou náročností jednotlivých typů úkolů. Pracovní listy mohou být prostředkem, který vede k rozvoji těchto znalostí a dovedností.

OSNOVA: Úvod; Podklady pro tvorbu pracovních listů; Tvorba pracovních listů; Zkouška pracovních listů a vyhodnocení; Závěr.

Seznam doporučené literatury:

LEPIL, Oldřich. Teorie a praxe tvorby výukových materiálů. Olomouc: Univerzita Palackého, 2010. ISBN 978-80-244-2489-7.

VANĚČEK, David. Didaktika technických odborných předmětů. Praha: ČVUT, 2016. ISBN 978-80-01-05991-3.

Gavora, Peter, 1942-. Úvod do pedagogického výzkumu. Peter Gavora

Microsoft, Průvodce programování v C#, https://docs.microsoft.com/cs-cz/dotnet/csharp/programming-guide

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Prof. RNDr. Emanuel, Svoboda, CSC., MÚVS ČVUT v Praze, Oddělení pedagogických a psychologických studií

Jméno a pracoviště konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: 12. 12. 2019 Termín odevzdání bakalářské práce: 30. 4. 2020

Platnost zadání bakalářské práce: 23. 9. 2021

Podpis vedoucí(ho) práce | Podpis vedoucí(ho) ústavu/katedry | Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

31. 3. 2020

Datum převzetí zadání

O. T. Klimeš

Podpis studenta(ky)

KLIMEŠ, Ondřej Tomáš. Učební text pro výuku odborného předmětu. Praha: ČVUT 2020. Bakalářská práce. České vysoké učení technické v Praze, Masarykův ústav vyšších studií.

MASARYKŮV ÚSTAV
VYŠŠÍCH STUDIÍ
ČVUT V PRAZE

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně. Dále prohlašuji, že jsem všechny použité zdroje správně a úplně citoval a uvádím je v přiloženém seznamu použité literatury.

Nemám závažný důvod proti zpřístupňování této závěrečné práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Praze dne: 05. 05. 2020

Podpis:

Ing. Ondřej Tomáš Klimeš

Digitálně podepsal Ing. Ondřej Tomáš Klimeš
Datum: 2020.05.05 19:51:35 +02'00'

Poděkování

Děkuji vedoucímu této bakalářské práce prof. RNDr. Emanuelu Svobodovi, CSc., za usazení práce do myšlenkového rámce, za rady a komentáře v průběhu psaní práce a za možnost nahlédnout do práce poctivého pedagoga. Stejně tak děkuji PhDr. Blance Jirkovské, Ph.D. za ochotné převzetí vedení této bakalářské práce po zesnulém panu Svobodovi a za komentáře a rady, které přispěly k finalizaci této práce. Dále děkuji své ženě za pomoc se srozumitelností této práce a jazykové korektury.

Abstrakt

Tématem práce je vytvořit učební texty pro výuku programového vybavení – C#, 3. ročníku SPŠE Ječná. Cílem práce je hledat efektivní způsob výuky, který by žákovi předal nejen znalosti, ale také dovednosti. Teoretická část je zaměřena na specifika výuky programování, od volby cílů, výukových metod, až po pedagogickou diagnostiku. Praktická část je zaměřena na tvorbu učebních textů a jejich napojení na výukové metody.

Klíčová slova

učební text, výuka, programování, C#

Abstract

The theme of this work is to create a textbook for the teaching of programming resources – C# at the SPŠE Ječná, which is taught in the 3rd year of study. The search of effective teaching, which learns students not just knowledge, but also skills, is a goal of this thesis. Theoretical part is focused on specifics of programming teaching, from defining goals, teaching methods, to pedagogical diagnostics. Practical part is focused on creation of the textbook and its connection to teaching methods.

Key words

textbook, teaching, programming, C#

Obsah

Slovník pojmů

ASP.NET	Platforma pro vývoj webových aplikací v .NET.
ADO.NET	Platforma pro komunikaci s databázovým serverem v .NET.
breakepoint	Breakepoint je značka, kterou programátor vkládá na místo, kde se má běh programového kódu zastavit. Slouží ke zrychlení procesu odladění programu.
C#	Vysokoúrovňový objektově orientovaný programovací jazyk (Microsoft)
connection string	informace potřebné k připojení program k databázovému serveru
Domain Name System (DNS)	Systém evidence názvů členů v doméně počítačové sítě.
File Transfer Protocol (FTP)	Komunikační protocol v počítačové síti, určený k přenosu souborů.
GET	Metoda dotazu na webový server.
Hypertext Transfer Protocol (HTTP)	Internetový protokol pro komunikaci s webovými server.
Internet Protocol (IP)	Komunikační protokol počítačové sítě.
namespace	Namespace je obdoba souborových složek, která slouží k organizaci (zařazení) kódu.
SQL	Standardizovaný strukturovaný dotazovací jazyk v relačních databázích.
timeout spojení	Časový interval, po kterém je ukončeno čekání na odpověď.
Uniform Resource Identifier (URI)	Webová adresa, která identifikuje server, službu a parametry.
User Datagram Protocol (UDP)	Komunikační protokol v počítačové síti. Datagram, bez ověření doručení.

Úvod

Výuka programování, jako podobor IT technologií, je aktuálním tématem, neboť současná společnost se stále více zaplétá do informačních vazeb. Těžko bychom hledali obor, který se dnes obejde bez IT technologií, a to nejen ve formě rychlého zdroje informací, ale stále více ve formě efektivního pomocníka. Masové rozšíření IT technologií přináší potřebu nemalého počtu školených osob, které budou IT služby zajišťovat. Na jedné straně je v dnešní době možnost studia IT technologií snadno dostupná, ale na druhé straně didaktika hromadné výuky programování je na začátku své cesty. Výuka předmětů, jako jsou matematika, fyzika, nebo jazyky mají málo proměnlivý obsah a dlouhou historii, ze které lze čerpat zkušenost, ověřené postupy, příklady a podobně. Naopak IT obory mají výrazně proměnlivý obsah s pouze nepatrnou historií - způsob jejich výuky je doširoka otevřenou problematikou.

Tato práce zpracovává problematiku tvorby pracovních listů v úzkém provázání na celkovou problematiku didaktiky zvoleného předmětu. Úzké zaměření pouze na tvorbu výukových listů by bylo neukotveným výstupem, jehož použitelnost by byla diskutabilní.

Cílem práce je vytvořit dílčí návrh výuky předmětu „Programové vybavení“ s ucelený pohledem do problematiky tvorby výukových cílů, forem výuky, diagnostiky až po zpracování vybraných konkrétních témat.

Práce využívá řadu zdrojů, kde těmi hlavními byly znalosti a dovednosti autora získané během studia „Učitelství odborných předmětů“ a zkušenosti získané během více než dvacetileté praxe v oblasti vývoje informačních systémů a tříleté praxe učitele informačních technologií.

TEORETICKÁ ČÁST

1 Specifika výuky programování

Programování, coby tvůrčí proces, obsahuje prvky analýzy zadání, algoritmizace, kódování a ověřování. Také se často užívá experimentování, napodobování, přepisování kódu třetích stran. Didaktika oboru tedy vyžaduje řadu specifických přístupů, které je na jedné straně možno převzít z výuky podobných předmětů (například matematiky), ale také vyžaduje nalézt svébytné techniky vytvořené na míru tomuto oboru.

1.1 Stanovení cílů výuky

Vývoj informačních systémů (IS) je obor, který se rozmáhá vysokým tempem. Existuje řada konkurenčních přístupů, vývojových prostředí a programovacích jazyků. Z didaktického pohledu je třeba zvolit vhodnou strategii výuky, kterou může být:

- vyučovat obecné (/teoretické/principiální) poznatky vývoje IS
- věnovat se vhodně zvoleným platformám ve vývoji IS v konkrétním programovacím jazyku a vývojovém prostředí (například platforma ASP.NET core v jazyku C# ve vývojovém prostředí Visual Studia)
- věnovat se technikám, jak si osvojit konkrétní vývojovou platformu – učit se učit (zaměřit se na dohledávání informací a práci s nimi - například z online zdrojů)

Stanovení poměrů zastoupení

Pro naplnění Rámcové vzdělávací programy (RVP) (NPIČR, 2020) je třeba ve vhodném poměru zahrnout všechny tři způsoby, kdy stanovení konkrétního poměru bývá nemalým problémem. Důležitými aspekty volby jsou:

- profilace školy: vedení školy může například orientovat výuku na obecné poznatky, nebo praktické (rutinní) dovednosti v konkrétním programovacím jazyce
- počet odborných (počítačových) učeben: teoretické základy lze vyučovat v běžné učebně pro celou třídu (tj. jedna učebno-hodina na třídu), naopak cvičení je třeba provádět v odborných učebnách s polovinou třídy (tj. dvě učebno-hodiny na třídu)
- počet a specializace učitelů: pokud mám učitele, který se specializuje na programování v jazyce Java, pak může učit obecné poznatky, ale ne programování v jazyce C#

Vliv skladby žáků

Dalším důležitým aspektem je skladba žáků. Pokud vyjdeme ze statistik, pak počet škol nabízející vybrané technické maturitní studijní obory znázorňuje Tabulka 1 (NPIČR, 2020).

Tabulka 1: Přehled počtu škol pro vybrané technické maturitní studijní obory

Kód oboru	Název oboru	Počet škol v ČR ve školním roce 2019/2020
18-20-M/01	Informační technologie	73
23-41-M/01	Strojírenství	40
26-41-M/01	Elektrotechnika	37
36-47-M/01	Stavebnictví	20

Zdroj: vlastní zpracování

Pokud vezmeme v úvahu, že před dvaceti pěti lety byl počet škol vyučující informační technologie v řádu jednotek, pak je patrné, že studium informačních technologií se stalo „populárním" oborem[1].

Široká nabídka možností studia přináší široké spektrum žáků, kdy nelze očekávat, že většina žáků se o obor aktivně zajímá. Je proto potřeba do výuky zařadit i elementární základy (například, co je to příkazový řádek) a více se zaměřit na získávání praktických dovedností, neboť nelze očekávat, že žák se oboru aktivně věnuje mimo školu. Úzká specializace na obecné principy by pak mohla vést k formálnímu vzdělání.

Podpůrné nástroje

Moderní vývojová prostředí jsou vybavena řadou podpůrných nástrojů, které výrazně zjednodušují kódování a ověřování funkčnosti programového kódu. Jedním z těchto nástrojů je, že prostředí „napovídá" jména objektů, jejich vlastností a metod. Například v případě Visual Studia se tento nástroj nazývá IntelliSense (VS-intellisense, 2020). Tyto nástroje výrazně ovlivňují, co musí vývojář znát a s jakou přesností (co musí „nosit v hlavě"). Pokud tyto nástroje vývojář k dispozici nemá (například programování v notepadu), pak musí u frekventovaných objektů znát jejich přesné názvy, stejně tak jejich metod a vlastností, neboť jejich vyhledávání v dokumentaci není efektivní. Pokud vývojář má k dispozici tyto nástroje, pak stačí, aby si pamatoval pouze částečný název, nebo i jen počáteční písmeno, neboť během psaní programového kódu mu nástroj nabízí relevantní seznam možností. Další vlastností tohoto nástroje je, že v případě výjimky nabízí možná řešení (například chybějící using namespace).

Na využití těchto nástrojů lze z didaktického hlediska nahlížet tak, že:

- „precizní" zapamatování nemusí předcházet použití: žák může pracovat na programovém kódu, i když zná potřebné názvy jen částečně.

[1] Zkušenost autora je, že výběr oboru často nevychází ze zájmu žáka o programování či správu informačních systémů, ale ze zájmu o hraní počítačových her, nebo sdílení obsahu přes internet. Studium často přináší rozčarování, neboť žák nemusí být svými vlohami predisponován pro toto studium.

- ✓ Žák může využít svou paměťovou „kapacitu“ k zapamatování většího množství „nepřesných“ informací.

- ✗ Využití nástroje bez nezbytného pochopení problematiky může vést k netypickým a obtížně řešitelným stavům: například, pokud v případě výjimky, která se zobrazí z důvodu chybného názvu objektu (typicky zaměněné velká a malá písmena, překlepu), žák namísto opravy názvu objektu využije tento nástroj k vytvoření nového objektu. Žák nebývá schopen problém vyřešit a pro učitele také bývá náročné „rozluštit“, co problém způsobuje.
- ✗ Žák namátkou volí možnosti z nabídek, bez potřebného promyšlení.

Je otázkou osobních preferencí učitele, jak dalece promítne tyto možnosti do výuky a diagnostiky.

Orientace na obecné poznatky vývoje IS

Výuka obecných poznatků vybaví žáka teoretickými nástroji, jak úlohu řešit. Způsob aplikace návrhového vzoru v konkrétním jazyce, nebo vývojovém prostředí buď navazuje v další výuce, nebo může být předmětem samostudia.

Mezi obecné poznatky vývoje IS můžeme zařadit:

- algoritmizace (Algoritmus)
- návrhové vzory (EN design pattern) (Pattern)

Shrnutí:

- ✓ počet frekventovaných návrhových vzorů není velký (v řádu jednotek)
- ✓ většina jazyků umožňuje návrhové vzory implementovat
- ✓ lze pracovat na úrovni zapamatování (dle Bloomovy taxonometrie) (Vaněček, 2016)

- ✗ žák si obtížně propojí teoretický poznatek s konkrétní implementací
- ✗ problém s motivací žáků

Orientace na vybrané platformy, programovací jazyky, vývojové prostředí

Bez vytváření konkrétního programu by výuka programování byla jen těžko realizovatelná. Práce na konkrétním programu žákovi zpřístupní i dovednosti a zkušenosti, které by při teoretické výuce byly jen těžko „uchopitelné“. Například se jedná:

- čas potřebný na kódování aplikace
- organizace pracovní plochy (kde bude editor kódu, kde budou složky projektu, jeho vlastnosti a podobně)
- získání informací o chybách kódu (syntaktických i sémantických) a způsob jejich odstranění

- jak přenést rozpracovanou práci mezi počítači (například ze školního počítače na osobní počítač žáka)

Pokud se při výuce bude psát konkrétní program, je třeba vhodně zvolit:

- programovací jazyk
- vývojové prostředí
- platformy/rozhraní

Při výběru je třeba zohlednit aspekty, jako:

- potřebný čas pro „zaučení" žáka v používání vývojového prostředí
- možnost demonstrace obecných poznatků (například, pokud je do výuky vybrán MVC návrhový vzor, mít možnost jej prakticky vyzkoušet)
- dostupnost pro žákovo osobní užití (například, aby nebyly zpoplatněny)
- množství pracovních míst, kde kvalifikačním kritériem je zvolená kombinace
- velikost komunity, která se zvolenou kombinací zabývá (především pro možnost dohledat informace z webu)

Shrnutí:

- ✓ žák získá dovednosti, které může přímo uplatnit v praxi
- ✓ žák se může věnovat oboru i mimo školní prostředí
- ✓ zaměření na dovednosti dobře vyvažuje ve výuce často dominující zaměření na znalosti
- ✗ obtížnější diagnostika znalostí
- ✗ vysoké technické nároky a potřeba odborné správy informačních systémů (nezbytnost přítomnosti IT správce ve škole)

Učit se učit

Vysoká dynamika oboru přináší též nejistotu, zda znalosti a dovednosti, které žák získá ve výuce, budou uplatnitelné v praxi (zda mezi tím nezastarají). Na druhou stranu vysoká dynamika oboru vyžaduje sledovat trendy a přizpůsobovat výuku tak, aby žák přicházel na pracovní trh se znalostmi a dovednostmi, které se budou v daném období poptávat. Nelze tedy tomuto riziku čelit tím, že se zaměříme pouze na „obecné", neboť s ohledem na vysoký počet absolventů jsou poptávány specifické dovednosti.

Příkladem krátkého života „nadějné" vývojové platformy může být aplikační platforma Microsoft Silverlight (Silverlight, 2020). Tato platforma byla uvedena v roce 2007 a společností Microsoft podporovaná a preferovaná. S ohledem na celosvětový vývoj v oblasti webových aplikací, kdy se zvolená koncepce ukázala jako nevhodná, byl její další vývoj v roce 2012 ukončen a její podpora skončí v roce 2021.

Možným řešením je zahrnout do výuky rozvoje kompetencí k dalšímu vzdělávání. Toto zahrnuje nácvik techniky, jak postupovat při zvládnutí nové vývojové platformy/ programovacího jazyka, stejně tak orientace ve zdrojích informací a jejich vyu-

žití. Možnostmi, jak si osvojit novou vývojovou platformu/programovací jazyk, například jsou:

- najít si vhodné zadání a zkusit si napsat aplikaci na dané platformě/programovacím jazyku: možno i s finančním ohodnocením, které na jedné straně zvyšuje motivaci a na druhé straně dodá „tlak" na výsledek
- studium ukázkové aplikace: softwarové společnosti, které uvádějí novou platformu/ programovací jazyk, nabízejí řadu ukázkových aplikací, které demonstrují jejich funkcionalitu a způsob užití
- studium dokumentace, dotazy na komunitu, rekvalifikační kurzy a podobně: tento postup je obdobou studia na škole

Do výuky je možné zařadit prvky, které žáka povedou k získání a prohlubování potřebných dovedností. Například je možné:

- učitel zadá žákovi vývoj aplikace dle konkrétního zadání (projektová výuka): žák pro její zdárné zvládnutí potřebuje získat celou řadu dalších znalostí a dovedností. Vytvoří si tak „užitečnou" vazbu mezi prací na vývoji aplikace a učením se dané platformy/programovacího jazyka
- učitel v hodině analyzuje a komentuje programový kód, který vytvoří sám, nebo jej (i před žáky) vyhledá na webu: žák si osvojí techniku, jak vyhledávat a procházet „cizí" programový kód
- učitel nechá žáka samostatně „dohledat" potřebné informace: například pokud žák potřebuje pomoc s problémem, učitel mu poradí pouze název objektu s doporučením, aby si jeho použití dohledal
- učitel nezodpovídá všechny dotazy, ale odkazuje žáky mezi sebou: žák se tak učí využívat znalostí komunity

Shrnutí:

- ✓ žák získá dovednosti, které bude potřebovat po celou dobu své pracovní kariéry
- ✓ rozšíří si spektrum zdrojů informací
- ✗ žáka může vést k „lenosti", kdy pro něj bude jednoduší řešení vyhledat, než jej sám nalézt
- ✗ speciálně webové zdroje nabízejí veliké množství často rozdílných řešení pro danou úlohu, že mohou žáka „zahltit" a znejistit

Ukázka volby cílů

Volba cílů primárně vychází z RVP 18 – 20 – M/01 Informační technologie (Ministerstvo školství, 2008) a příslušný školní vzdělávací plán k předmětu PV, cvičení programování C#.

Tato práce je orientována na užití:

- programovací jazyk: C#
- vývojové prostředí: Visual Studio 2019 v jazykové verzi EN[2]

Výukové cíle mohou být nastaveny následovně (nejedná se o ucelený koncept).

Obecné cíle

žák:

- ovládá syntaxi programovacího jazyka
- vytváří jednoduché aplikace a dokáže je zprovoznit
- vhodně komentuje a popisuje programový kód

Specifické dovednosti:

žák se ve vývojovém prostředí:

- orientuje v jednotlivých částech projektu
- používá nástroje pro zjištění informací o objektu, nebo jeho vlastnos-tech/metodách
- provádí ladění programového kódu (například krokování kódu, vkládání a mazání breakepoint)

žák dovede pro danou úlohu:

- zvolit vhodný typ projektu a založit jej
- nastavit vhodný namespace

žák ovládá základy vývojové platformy

- konzolová aplikace
- desktopová aplikace
- webová aplikace

1.2 Formy výuky

Motivace žáka se stala dominantním tématem v přípravě na výuku (Hrabal & Pavelková, 2010). Na jedné straně je v dnešní době programování atraktivním tématem, na druhé straně výuka programování čelí řadě komplikujících faktorů. Jedná se o následující:

- *programování je vysoce abstraktní činnost*: analýza úlohy a její algoritmizace je duševně náročným úkolem, který se promítá do rychlé únavy žáka

[2] Autor doporučuje požívat anglickou jazykovou mutaci všech vývojových prostředí. Anglicky korespondující komunita je dominující a tak se usnadní vyhledávání odpovědí na celou škálu otázek, souvisejících s výjimkami, doporučenými postupy a podobně.

- *výuka probíhá v dvojhodinovém bloku*: je třeba předcházet únavě žáka vhodným střídáním činností a vjemů
- *práce na počítači je visuálně a motoricky náročná*: během programování musí žák provádět řadu precisních motorických úkonů (například pozicování kurzoru myši), visuálně procházet kód, přecházet mezi různými prostředími (psaní programového kódu, jeho ladění a podobně), což se promítá do zvýšené únavy žáka.
- *žák je při práci na počítači ve strnulé fyzické pozici*: nároky na preciznost při práci s počítačem vedou ke strnulému pozici žáka za počítačem. Žák často o přestávce využívá počítač k duševní relaxaci (například k prohlížení obsahu webu), ale tím neprospívá k fyzické relaxaci.

Udržet žákovu pozornost a také předcházet různým zdravotním komplikacím je částečně možné „pestrostí" výukových forem. Vhodná kombinace forem by měla splňovat:

- přiměřené dávkování „problémů": zvolit formu, kterou provází „přiměřené" množství nechtěných výjimek a jiných komplikací
- střídání duševních činností: žák je střídavě naladěn na příjem a výdej informací, například po krátkém výkladu (cca 2 minuty) si žák novou informaci vyzkouší (cca 5 minut)
- zahrnout formy, které vyžadují fyzický pohyb: například na teoretický výklad, nebo opakování se lze přesunout do běžné učebny. Tato technika krom jiných výhod přinese, že žáci získají krátký oddychový čas mezi výkladem a začátkem práce na úkolu, který je navíc doprovázen pohybem. Další možností je „trestat" žáky za nesplnění domácího úkolu dřepy, nebo kliky, nebo zahrnout do hodiny rozcvičku.
- různé vizuální podněty: zvláště pro oči je důležité střídat pohled na krátkou vzdálenost s pohledem na delší vzdálenost. Jednou z možností je v cca 20 minutových intervalech požádat žáky, aby se podívali na učitele, který jim řekne nějaký komentář (kterých bývá dostatek)

Některým specifickým formám výuky se věnují následující podkapitoly.

Mikro výklad – dílčí krok

Vysoká abstraktnost úloh v kombinaci s vysokou komplexitou bývá „spolehlivým" důvodem, že žák přestává úloze rozumět. Bez porozumění žák neví, jak dál pokračovat, přestává pracovat, nebo jen mechanicky opisuje z předlohy (je-li k dispozici). Možný východiskem je pracovat u žáka s parciálním porozuměním (pochopením na „krátkou" vzdálenost), které je potřebné k provedení dílčího kroku. Syntézu těchto parciálních znalostí a dovedností přesunout do opakování, případně dalšího procvičování. Při takovémto postupu je efektivní, aby žák mohl bezprostředně přenést získané informace do konkrétní činnosti.

Shrnutí:

- žák nemusí rozumět celé problematice úlohy
- postačí krátká pozornost žáka na výklad učitele
- vhodně se střídá typ zátěže žáka (příjem informací a jejich aplikace)
- synchronizuje se práce žáků: učitel získá kontrolu nad tím, že všichni žáci zvládli klíčovou část úlohy

- „rychlejší" žák nepracuje svým tempem

Příklad užití postupu mikro výklad – dílčí krok

Úloha, ve které se procvičuje síťová komunikace UDP, vyžaduje od žáka základní znalost a orientaci ve značně rozsáhlé problematice. Předmětem úlohy bývá procvičit jak odesílání, tak příjem dat, ke kterému se používají stejné programové prostředky, jen s jinými kombinacemi parametrů a metod. Řešení úlohy pro příjem vyžaduje několik postupných kroků:

- vytvořit socket (přípojku) pomocí vhodné kombinace tří parametrů
- vytvořit endpoint (koncový bod komunikace) pomocí kombinace IP adresy a portu
- propojit socket a endpoint (bind)
- vytvořit cyklus, ve kterém se budou přijímat data (socket receive) a případně zpracovávat (například vypsat na konzoli)

Náročnost úlohy lze spatřovat v těchto aspektech:

- nevhodná kombinace parametrů socket se projeví až v momentě přijímání dat (o několik programových řádků dál)
- hlášení o výjimce jsou pro žáka těžko „rozluštitelná": například „Došlo k pokusu o přístup k soketu způsobem zakázaným jeho přístupovými oprávněními.", nebo „Požadovaná adresa není v tomto kontextu platná."
- pro žáky bývá náročné zjistit IP adresu koncového bodu, dále nemívají povědomí o významu portu
- procvičování úlohy se může dostat do kolize s pravidly místní (školní) počítačové sítě: k úloze se využívá školní počítačová síť, která obvykle bývá zabezpečena různými pravidly

Řešení pomocí postupu mikro výklad – dílčí krok

Učitel uvede žáky do problematiky vhodnou formou (například inscenační). Následuje řešená forma, kdy všichni žáci postupují společně:

- učitel před každým programovým řádkem vysvětlí, k čemu slouží (naváže jej na předchozí výklad), vysvětlí význam všech parametrů a jejich potřebných kombinací (cca 1 minuta)
- žáci na pokyn učitele napíší programový řádek (cca 1-5 minut)
- žák ověří funkčnost programového kódu (pokud je to již možné)
- učitel sleduje práci žáků, případně individuálně pomůže s řešením

- učitel se ujistí, že všichni žáci napsali požadovaný řádek a že je funkční

Obdobně se postupuje až do zvládnutí „kritické " fáze programového kódu. Následně mohou žáci pokračovat individuálním tempem.

Eliminace výjimek v komplexních úlohách

Komplexní programové úlohy (například komunikace s databázovým serverem) jsou zdrojem natolik rozmanitých výjimek, že žák není schopen je samostatně řešit. Cvičení je proto vhodné koncipovat tak, aby se snížilo množství „rušivých" výjimek, neboť jejich sekvence může žáka demotivovat, nebo odradit od další práce. Možností je ověřit si jednotlivé parametry pomocí nezávislých a ověřených nástrojů, tedy nástrojů, které řeší stejnou úlohu. Žák tím získá důvěru k části kódu a důvody případných výjimek bude hledat v menším spektru variant.

Shrnutí:

- ✓ žák je méně vystaven „bezradné" situaci: může mít pozitivní dopad na psychiku žáka
- ✗ žák se musí seznámit s aplikací k ověření parametrů

Příklad komplexních výjimek

Příkladem komplexity je, když při volání příkazu SqlCommand nastane výjimka „*ExecuteNonQuery requires an open and available Connection. The connection's current state is closed*.". Žák vlastní úvahou, vyhledáním či dotazem na učitele/spolužáka zjistí, že neotevřel spojení na databázový server. Přidá tedy programový řádek, ve kterém otevírá spojení. Následně může nastat jiná výjimka, která může být způsobena (výčet není kompletní):

- cesta k serveru je neplatná
- server je nedostupný (vypnutý)
- nejsou ve firewallu otevřené komunikační porty
- databáze neexistuje
- databáze je blokovaná
- uživatel neexistuje
- uživatel nemá práva k databázi
- heslo je neplatné

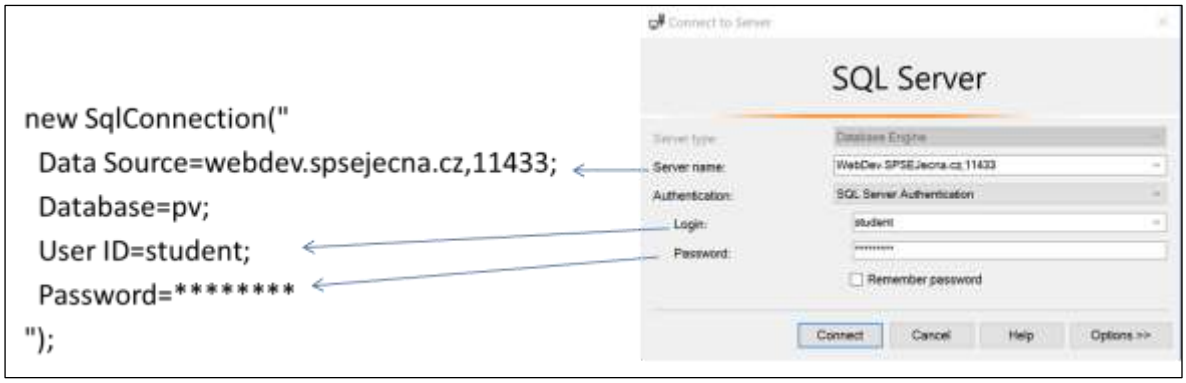
Proces zprovozňování programového kódu navíc komplikuje, že výjimky se zobrazují postupně (po odstranění jedné se zobrazí další).

Řešení (eliminace) množství výjimek

K ověření spojení se serverem může žák použít analytické nástroje databázového serveru. Tyto nástroje se potřebují, obdobně jako programový kód, připojit k databázovému serveru se stejnými parametry. Pokud se žák připojí pomocí para-

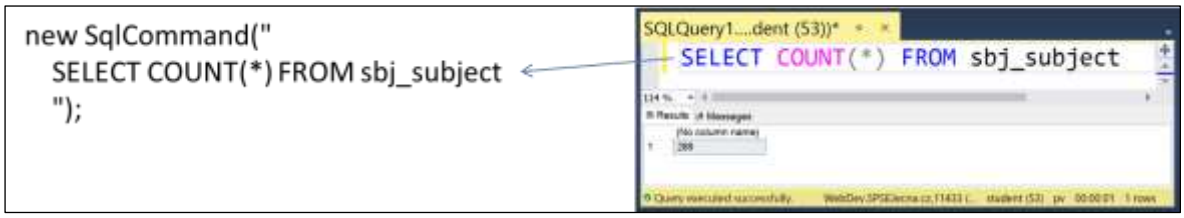
metrů k databázovému serveru v takovémto nástroji, pak budou fungovat stejné parametry i v jeho programovém kódu (viz Obrázek 1 a Obrázek 2)

Obrázek 1: Ověření správnosti parametrů instance SqlConnection pomocí aplikace Microsoft SQL Management Studio



Zdroj: vlastní zpracování

Obrázek 2: Ověření SQL příkazu pomocí query v aplikaci Microsoft SQL Management Studio



Zdroj: vlastní zpracování

Řízené vyvolávání výjimek v programovém kódu

Vývoj aplikací je úzce spjat s jejich testováním a řešením provozních problémů. Je vhodné tedy žáka vybavit nejen dovedností vytvořit aplikaci, ale také řešit situaci, kdy aplikace nefunguje.

Při zprovozňování aplikace bývá sice žák nucen řešit výjimky, ale žák je v daný moment vnímá spíš rušivě (neboť jej „zdržují" od zprovoznění aplikace). Navíc se většinou jedná o problémy se syntaxí, chybějící knihovny (dll library) či odkazy na příslušné jmenné prostory (namespace). Tyto chyby brání kompilaci kódu, tedy bez jejich odstranění není možné kód ani spustit. Nejedná se tedy o výjimky, které mohou nastat během provozu aplikace. Řešení těchto výjimek je spíš „vynucenou" dovedností.

S provozními výjimkami se žák může seznámit prostřednictvím řízeného vyvolání výjimky. Jedná se o techniku, kdy ve funkční a ověřené aplikaci žák cíleně změní některý z parametrů a zkouší, jak na to bude aplikace reagovat. Cílem činnosti žáka je tedy „objevit" výjimku, která souvisí s konkrétním chybným parametrem.

Shrnutí:

- žák pracuje formou objevitelské úlohy

- žák se učí nahlížet na výjimku jako na užitečný nástroj programátora (tedy nejen zdržující, či obtěžující)

Příklad cíleného vyvolání výjimky

Příkladem řízeného vyvolání výjimky je použití neplatné URI adresy při HTTP komunikaci. K HTTP komunikace lze použít objekt HTTPClient (Microsoft, 2020). Komunikace však vyžaduje několik postupných kroků:

- sestavení URI adresy
- vytvoření instance objektu HTTPClient
- vyvolání některé z metod typu Get
- zpracování návratové hodnoty z předchozí metody

Možné výjimky při komunikaci zapříčiňuje (výčet není kompletní):

- syntaktická chyba v URI adrese
- chyba DNS
- chyba certifikátu (při HTTPS komunikaci)
- timeout spojení

Žák nejprve vytvoří programový kód a učitel se ujistí, že programový kód funguje. Následně vyzve žáka, aby zjistil, jaké chybové hlášení se zobrazí, když:

- do URI adresy něco dopíše (zneplatní ji)
- změní HTTP na HTTPS (nebo naopak)
- změní HTTP na FTP
- sníží timeout komunikace na 0,1s

Programování „nanečisto“

Pokud se programátor seznamuje s novou technologií, novým objektem, nebo třeba s metodou, nejprve si napíše programový kód „nanečisto“. Programováním „nanečisto“ je míněno, že:

- program je minimálně parametrický: všechny potřebné argumenty jsou přímo vkládány do programového kódu
- program obsahu je pouze nezbytný kód pro zprovoznění úlohy: neřeší se ošetření výjimek, komentáře a podobně
- je-li možno, pracuje se přímo v zaváděcí metodě Main() (nevytvářejí se specializované objekty)

Poté, co se s úlohou seznámí, naprogramuje ji „načisto“ a integruje do nového/stávajícího programového kódu. Programováním „načisto“ je míněno, že programový kód je:

- „rozumně“ zobecněn, aby byl použitelný i pro obdobné úlohy (nejen tu, která se právě řeší)

- vhodně strukturován, v případě objektového programování do tříd, metod a vlastností
- vytvořen tak, že proměnlivé složky jsou parametrizovány (přihlašovací údaje, cesta k serveru a podobně)
- ošetřen na výjimky, které mohou při volání nastat
- doplněn o komentáře pro lepší čitelnost a pochopení

Pokud je tato technika užitečná v praxi a stejně tak splňuje didaktická pravidla od konkrétního k obecnému, je užitečné ji integrovat do výuky.[3]

Shrnutí:

- ✓ snižuje komplexitu na minimální úroveň, kód je lépe pochopitelný, ověřitelný
- ✓ je možné použít příklady dostupné například z webu
- ✓ pro jednoduchost se snáze případně získá pomoc od kolegy, z webu
- ✓ minimalizuje časové nároky na zvládnutí úkolu
- ✓ jednoduše jde přenášet „z počítače na počítač", kdy jeden student jej může poskytnout jinému
- ✗ v hodinách nebývá dostatek času na přepsání kódu načisto, žák se ve škole málo setká s „čistým" kódování

Příklad programování nanečisto

Připojení k databázovému serveru není triviální úlohou, byť díky specializovaným knihovnám je potřebný programový kód minimální. Přesto prvotní seznámení nebývá jednoduché. K procvičení látky je účelné napsat nejprve kód nanečisto (příklad viz Obrázek 3).

Obrázek 3: Ukázka kódu pasného nanečisto

```
SqlConnection pripojeni = new SqlConnection("Application Name=testSQL; ...");
pripojeni.Open();
SqlCommand vyber = new SqlCommand("SELECT id FROM sbj_subject WHERE id=1", pripojeni);
vyber.ExecuteScalar();
```

Zdroj: vlastní zpracování

Programový kód je prost komentářů, proměnlivé složky (pro názornost podbarveny oranžově) jsou součástí kódu a nejsou ošetřeny výjimky. Nic z toho nebrání kód vyzkoušet a pochopit jeho fungování. Možno říct, že má i didaktickou názornost, kde žák přímo vidí, kam se vkládá SQL příkaz, kam se vkládá connection string a podobně.

[3] Autorova zkušenost je, že tematické plány programování bývají naddimenzovány, od snahy žáka co nejvíce naučit až po vytváření dojmu. Často proto zbývá čas pouze na psaní nanečisto. Napsat kód načisto může být dobrým domácím úkolem k tématu.

Po zvládnutí úkolu je možné stejný programový kód přepsat načisto (příklad viz Obrázek 4).

Obrázek 4: Ukázka kódu pasného načisto

```
class DataControler
{
    //Vlastnost nesoucí informace o připojení k databázovému server
    public static string ConnectionString { get; set; }

    //Vrátí database connection
    public static SqlConnection GetConnection()
    {
        if (String.IsNullOrEmpty(ConnectionString)) throw new Exception("ConnectionString nesmí být přázdný.");

        return new SqlConnection(ConnectionString);
    }

    //vrátí skalární hodnotu na základě sql příkazu
    public static object GetScalar(string sqlScript)
    {
        if (String.IsNullOrEmpty(sqlScript)) throw new Exception("SQL příkaz nesmí být přázdný.");

        using (SqlConnection pripojeni = GetConnection())
        using (SqlCommand vyber = new SqlCommand(sqlScript, pripojeni))
        {

            try
            {
                pripojeni.Open();
                return vyber.ExecuteScalar();
            }
            finally
            {
                if (pripojeni.State == ConnectionState.Open) pripojeni.Close();
            }
        }
    }
}
```

Zdroj: vlastní zpracování

1.3 Diagnostika výuky programování

Diagnostika dovedností je důležitou fází vzdělávacího procesu, kde její užití v předmětu programování má svá specifika. Výuka programování se řadí mezi odborné předměty a jejím výstupem by krom teoretických znalostí měla být praktická dovednost. Cílem diagnostiky v řešeném předmětu je především zjistit úroveň praktických dovedností žáka.

Použití více diagnostických nástrojů umožní získat věrnější stav znalostí a dovedností žáka. Stejně tak lze získat představu, kde výuka „nefunguje“, jaké informace žáci „nezachytili“ a podobně.

Vliv prostorového uspořádání učebny

Možnost diagnostiky učitelem je silně závislá na prostorové organizaci učebny. Důvody lze spatřovat v tomto:

- žák svou práci provádí na počítači – nemůže ji přinést učiteli ukázat
- v případě, že učitel chce žákovi něco ukázat, pak potřebuje psát na jeho počítači (používat jeho klávesnici a myš). Má-li učitel předejít zdravotním komplikacím (například bolesti v zápěstí), pak je vhodné si při psaní sednout.

- pokud bude učitel při snaze se dostat k žákovi muset žádat jiného žáka (souseda) o „puštění", může učitelova činnost působit rušivě

Existují technické prostředky, které umožňují učiteli vzdáleně pracovat na počítači žáka. Jejich použití má následující aspekty:

- lze jej použít při libovolném uspořádání učebny
- učiteli umožňuje rychle „přecházet" mezi počítači jednotlivých žáků

- technicky náročné a na problémy náchylné řešení
- vyžaduje součinnost IT správce školy – nemusí být jednoduché zajistit
- chybí, nebo je omezen kontakt se žákem

Vhodné prostorové uspořádání učebny by mělo splňovat:

- dostatek prostoru pro pohodlný přístup učitele k žákovi a to včetně židle, kterou si učitel nese
- „dostupnost" všech žáků

Počítače podél stěn

Vhodným uspořádáním počítačů je, aby byly umístěny podél pravé/levé stěny a orientovány směrem ke stěně:

- žák je dobře dostupný
- učitel může efektivně pozorovat práci žáků, aniž by je „rušil"
- střed učebny je vhodným místem pro interaktivní styl výuky

- žáci se musejí na učitele otáčet, pokud jim chce něco sdělit
- žáci se musejí otáčet na projektor

Počítače v souvislých řadách

Další vhodný typ upořádání je v souvislých řadách s dostatečným prostorem pro pohodlný průchod:

- žák může snadno sledovat projektor i učitele
- třída má „frontální" ráz – učitel může přehlédnout všechny žáky (vidí jim do obličeje)

- procházení učitele může být rušivé – před žáky se „míhá" učitel

Diagnostická zkouška

Cvičení programování svou povahou nabízí učiteli dostatečný prostor provádět diagnostické zkoušení. Jedná se především o tyto aspekty:

- cvičení probíhá v dvouhodinovém bloku – bývá dostatek času
- množství programového kódu nebývá velké (cca 5-20 řádek) - učitel se snadno zorientuje v žákově práci
- programový kód se píše v počítači - je pro učitele snadno čitelný (oproti ručnímu zápisu)
- žák často řeší chybové stavy, které nedokáže řešit a sám volá učitele pro radu - nevnucený příchod učitele za žákem

- nezřídka se vyskytují technické potíže, které učitel musí řešit, což bývá často zdlouhavé – žák ztrácí zájem o práci

Této skutečnosti může učitel využít k diagnostickému pozorování žáka, nebo k jeho orientačnímu zkoušení.

Průběh diagnostické zkoušky/pozorování

Možný průběh diagnostické zkoušky/pozorování je:

- po zadání úlohy učitel vyčká cca 5-10 minut, než se žáci „zaberou" do práce
- učitel projde třídu a zevrubně prohlédne práci žáků, aby na základě stavu rozdělané práce zjistil, zda žáci pochopili zadání a jsou schopni na zadání pracovat
- učitel případně může dovysvětlit/doplnit zadání
- učitel vytipuje žáka, který má největší potíže s úlohou, přisedne si k němu a návodnými otázkami mu pomůže započít úlohu – například mu zadá dílčí úkol. Následně může pokračovat k dalšímu žákovi.
- pokud většina žáků zdárně pracuje na úloze, může si učitel vytipovat žáka pro orientační zkoušení. Například si učitel nechá od žáka vysvětlit programový kód, zeptá se na teoretické souvislosti a podobně.
- učitel si vytipuje „zdatného" žáka, který už má úlohu v pokročilé fázi. Pročte si ji a polohlasně okomentuje jeho kód, pochválí jej a je-li možnost, ukáže mu cosi za hranicí běžné výuky.
- učitel též musí reagovat na dotazy žáků a pomáhat řešit technické potíže

Kvalifikační zkouška

Kvalifikační zkouška se obvykle provádí po dokončení jednoho až dvou tematických bloků (jsou-li v nějakém vztahu). Ke způsobu zadání lze použít většinu forem, záleží na učiteli, kterou formu preferuje.

Individuální ústní zkouška

Individuální ústní zkouška přináší „věrný" obraz o znalostech a dovednostech žáka. Při spojení individuální ústní zkoušky s praktickou zkouškou je možné také diagnostikovat specifické dovednosti, které se jiným způsobem těžko získávají.

Průběh zkoušky:

- žák si sedne k počítači učitele, učitel vedle něj
- učitel zadá žákovi úlohu (například vytvoř konzolovou aplikaci, která spočítá faktoriál čísla)
- žák komentuje, co se chystá provést
- během práce učitel klade žákovi doplňující otázky z teorie, které navazují na právě vytvářený kód (například: „proč jsi napsal public static")

- učitel pozoruje, zda žák užívá ladicí nástroje, případně jej k tomu vyzve (například „vlož do kódu breakpoint a spusť kód v krokovacím režimu“)
- učitel může návodnými otázkami pomáhat žákovi s prací na úkolu (například: „pokud potřebuješ provést opakovaně stejnou úlohu, jen s jiným číslem, co k tomu použiješ - cyklus“)

Praktická zkouška

Praktická zkouška je obdobná tomu, jak žák pracuje na vyučování. Zadáním je vytvořit program, který má specifikovanou funkcionalitu a případně je požadovaná i konkrétní forma (doplnit komentáře a podobně). Časové nároky bývají nejméně jedna vyučovací hodina.

K praktické zkoušce žák potřebuje počítač, jenž umožňuje:

- ✓ žákovi pracovat způsobem a v prostředí, ve kterém chceme diagnostikovat stav znalostí
- ✓ učiteli pozorovat u žáka pracovní postupy, schopnost řešit výjimky, ladění programu a podobně

- ✗ žákovi hledat informace na webu, lokálním, nebo síťovém disku
- ✗ žákovi vyměňovat informace, nebo část programového kódu se spolužáky

Při hromadném zkoušení touto formou se může stát, že u některých žáků spíše než programování se budou diagnostikovat jejich „managerské“ schopnosti. Technicky je sice možné nechtěné aspekty práce na počítači eliminovat, vyžaduje to však nemalou přípravu (přistupuje se k nim v případě praktické maturity). V běžné výuce je třeba zajistit požadované podmínky jinak (například zvýšenou kontrolou během zkoušky, porovnáním kódu žáků a odhalováním plagiátorství).

Odevzdání práce při hromadné zkoušce je náročnější než u ostatních forem, neboť výstupem praktické zkoušky je skupina souborů (tzv. projekt), kde žák nemusí buď omylem, nebo i záměrně odevzdat všechny.

Učitel musí zkoušku vyhodnotit na počítači, kde potřebuje značný čas na obnovu (zprovoznění) výstupu žáka. Nemůže tak provádět vyhodnocení například při cestě vlakem z práce, nebo v kabinetě (pokud nemá vlastní počítač).

Učitel se tedy dostává do situace, kdy aplikace vhodné formy zkoušení v případě hromadné zkoušky přináší tolik negativ, že její použití se stává nepraktické. Vhodná je především ve spojení s individuální ústní zkouškou (byť je časově náročná).

Písemná práce

Písemná práce je časově úsporným objektivním způsobem hodnocení a klasifikace, který má také řadu nevýhod, plynoucích především z chybějícího kontaktu s učitelem, jeho korekci/vedení (Vaněček, 2016).

Z obecného hlediska je využití písemné zkoušky v případě výuky programování obdobné, jako v jakémkoli jiném předmětu, podobně však má i svá specifika, například:

- ✓ žák je veden otázkami: například se nemůže „zaseknout" na zadání praktické zkoušky
- ✓ učitel může efektivně bránit opisování: žáci nemají k dispozici počítač a ani mobilní zařízení
- ✓ žáci tento způsob zkoušení „ovládají" z jiných předmětů

- ✗ způsob zkoušky je výrazně odlišný od práce na hodinách a prostředí, ve kterém vytvářejí programový kód
- ✗ některé dovednosti se tímto způsobem obtížně diagnostikují

Průběh zkoušky:

- učitel vyzve žáky, aby si vzali pouze psací prostředky a mobilní telefon
- učitel a žáci se přesunou do běžné učebny: běžná učebna nabízí učiteli lepší kontrolu
- učitel vyzve žáky, aby ponechali mobil na jeho stole: preventivní opatření proti opisování
- žáci si sednou do tří řad za sebou, v učebně zcela vpravo, uprostřed a zcela vlevo
- učitel vysvětlí žákům zadání a případně vztah mezi jednotlivými listy papíru: v případě, že k písemné práci je třeba ukázkový programový kód, učitel jej vytiskne na zvláštní papír
- učitel instruuje žáky, že pokud jsou hotovi, odevzdají vypracované zadání a přesunou se do odborné učebny: žáci mohou v odborné učebně diskutovat výsledky a neruší tak žáky, kteří ještě pracují na zadání[4].
- učitel rozdá žákům zadání
- po odevzdání od posledního žáka se učitel přesune zpět do odborné učebny a se žáky probere možné výsledky

[4] Na druhou stanu, odchod žáků z běžné učebny je sám o sobě rušivý a dále může v žácích, kteří ještě pracují na zadání, vyvolat nechtěný tlak, že by měli už končit. Je dobré žáky upozornit, že nemusejí chvátat. Fonty.

PRAKTICKÁ ČÁST

2 Komunikace s databázovým serverem

Jednou, v dnešní době základní, dovedností programátora je komunikace s databázovým serverem. Například, počet odpovědí vyhledávače Google na dotaz „program with Database Connection" vrací cca 483 000 000 odkazů.

Problematika výuky

Téma komunikace s databázovým serverem patří k náročnému učivu, důvody lze spatřovat v:

- znalosti SQL: pro zvládnutí tématu je potřeba zvládnout s porozuměním téma SQL script, který může být tématem stejného, nebo souběžného předmětu. Pokud žák SQL script neovládá, může sice kódovat spojení s databázovým serverem, ale pracuje převážně s SQL příkazy nabízenými (ukazovanými) učitelem.
- komunikace s databázovým serverem vyžaduje zapojení více programových objektů s řadou parametrů. Pokud je žák nezvládne správně nastavit, nedostaví se výsledek a žák má tendenci přestat pracovat (ztráta motivace).
- komplexnost tématu vyžaduje, aby žák dobře chápal princip tématu. Bez toho pochopení se snadno „ztratí" a buď přestane pracovat, nebo bude jen „kopírovat" ukázku učitele.
- k výuce je třeba též samotný databázový server, ať lokální (běžící na počítači žáka) či sdílený (přes počítačovou síť). Komplikované nastavení databázového serveru a firewallů, nebo problémy s výkonem počítačů může odrazovat samotného učitele od cvičení toho tématu (probere jej pouze teoreticky), nebo může narušit průběh cvičení.
- k efektivní práci žák musí též ovládat nástroje ke správě a analýze databázového serveru (například MS SQL Server Management Studio a SQL Server Profiler), které žákovi pomáhají řešit problémy spojené s testováním programové aplikace. Jejich zvládnutí není triviální.

Přínos zvládnutí tohoto tématu lze spatřovat v následujících bodech:

- žák disponuje základní dovedností pro tvorbu databázových aplikací
- žák získá zkušenost s komunikací klient-server a jejími specifickými problémy (timeout, nedostupnost serveru, ztráta konektivity, …)
- pokud jsou data, na kterých se komunikace zkouší, pro žáka atraktivní (například se nějakým způsobem žáka a jeho spolužáků týkají), pak žák si data „různě prohlíží" a tím bezděčně procvičuje téma.

Technologie a prostředí

Pro cvičení tématu bylo zvoleno:

- uživatelské prostředí Visual Studio (v aktuální verzi)
- programovací jazyk C#
- databázová komunikační platforma ADO.Net
- databázový sever Microsoft SQL Server Develop edition (v aktuální verzi)

Koncipování výuky

Tématu s ohledem na náročnost je vhodné věnovat dva dvouhodinové bloky. První blok je věnován volání a zpracování SQL příkazu na databázovém serveru bez, nebo s jednoduchým návratovým typem (non query, scalar). Druhý blok je věnován volání a zpracování SQL příkazu s hromadným návratovým typem, případně s parametrickým příkazem (data reader, data adapter, parameter).

Náročnost úlohy

Cvičení tohoto tématu vyžaduje, aby byl žákům dostupný databázový server s cvičnou databází. Server může být instalován na osobním počítači žáka (tzv. lokální), nebo může být žáky sdílený (tzv. síťový). Srovnání variant následuje.

Sdílení (síťový) server:

- společné přístupové údaje
- učitel může předem ověřit připojení k serveru
- učitel může předem připravit data
- učitel může lépe sledovat práci žáků
- možnost zpřístupnit databázový server i mimo konkrétní učebnu, nebo školu (žák může pracovat i z domova)

- žák může „destruktivním“ příkazem poškodit data, nebo databázi natolik, že znemožní práci ostatním žákům

Lokální databázový server:

- každý žák má částečně jiné parametry připojení k databázovému serveru, které znesnadňují „opisování“, resp. žák si může uvědomit jejich význam
- pokud žák provede „destruktivní“ příkazy (například smazání všech záznamů), neovlivní tím práci ostatních žáků

- potenciálně je zdrojem většího množství výjimek, jejichž řešení zaměstnává učitele
- učitel nemůže předem ověřit funkčnost připojení k databázovému serveru
- učitel má ztížený přístup k datům žáka
- učitel nemůže předem připravit ukázková data, resp. je komplikované a časově náročné data distribuovat

2.1 Připojení k SQL serveru a volání jednoduchého SQL příkazu

Cvičení má v žákovi prohloubit porozumění tématu a tvorbu dovedností potřebných k programové práci s databázovým serverem.

Specifické cíle

Žák:

- porozumí významu parametrů připojení (connection string) a sestaví jej pro konkrétní situaci (lokálního/síťového databázového server)
- úspěšně vytvoří spojení s databázovým serverem (SQLConnection) a otevře jej
- ošetří výjimky (exceptions), které mohou nastat při otevření spojení
- dokáže vytvořit SQL příkaz (SQLCommand) a nastavit jeho nezbytné vlastnosti (script a connection)
- úspěšně vyvolá SQL příkaz bez návratové hodnoty (non query), nebo s jednoduchým návratovým typem (scalar)
- ošetří výjimky (exceptions), které mohou nastat při volání příkazu

Opakování, prohloubení znalostí a motivace

Nezbytný programový kód potřebný k procvičení tématu není rozsáhlý (cca 10 programových řádek), tedy k jeho napsání a zprovoznění postačí jedna vyučovací hodina. Nabízí se tím možnost využít první hodinu dvojbloku k opakování a uvědomění si, jaké programové objekty budou potřeba a jaká je jejich role. S ohledem na množství času je zvolena *inscenační* metoda výuky.

Opakování má za cíl, aby si žák:

- uvědomil komunikující strany – klient : server
- uvědomil nezbytnost prostředníka – spojení mezi klientem a serverem
- uvědomil význam a určení tří nezbytných částí parametrů připojení – cesta k serveru, databáze, autentizace (data source, database, user id, password)
- uvědomil, že k vykonání příkazu je třeba spojení

Inscenační prostor a pomůcky:

- prostor s dostatečným místem před první řadou lavic[5]
- žák potřebuje papír, psací potřeby a mobilní telefon[6]

[5] Pokud je odborná učebna nevyhovující, pak je vhodné si předem vyhledat a zajistit vhodnou učebnu.

Zadání:

- žáci mají za úkol zhlédnout scénku a následně vytvořit algoritmus, který popisuje její průběh[7]

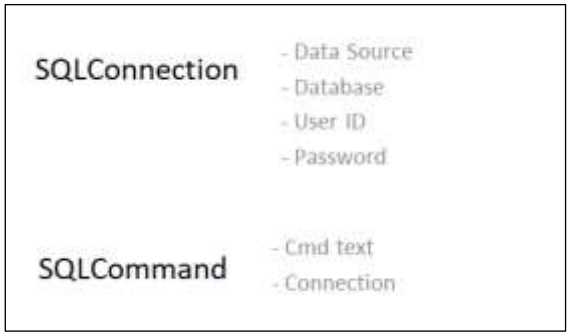
Role:

- třídní učitel
- žák, který se chce omluvit

Průběh:

- učitel vyzve dva žáky, aby zahráli scénku, ve které jeden z nich volá do školy třídnímu učiteli/učitelce, aby se omluvil, že z důvodu nemoci nepřijde do školy
- žák hrající žáka stojí při pravé stěně učebny
- žák hrající učitele stojí při levé stěně učebny
- ke komunikaci používají svůj osobní telefon
- v případě potřeby učitel zasáhne do scénky a nechá ji přehrát znova
- po skončení scénky učitel vyzve žáky, aby vytvořili algoritmus ke shlédnuté situaci
- během práce žáků na algoritmu si učitel připraví na tabuli informace (viz Obrázek 5)

Obrázek 5: Instruktážní text vlastností SQL spojení



Zdroj: vlastní zpracování

- učitel si projde žáky a prohlédne rozpracované práce. Snaží se najít různé přístupy (resp. jejich kombinace):
    - procesní: žák situaci rozdělí do dílčích kroků (bývá nejčastější)

---

[6] Budou potřeba dva mobilní telefony. Pokud nebudou k dispozici, resp. žáci je neposkytnou, pak je možné je nahradit třeba penálem, kalkulačkou.

[7] Týká se i žáků, kteří scénku předváděli.

    - signální: žák se zabývá přenosem informace, nikoli obsahem (středně zastoupená)
    - objektový: žák oddělí dva objekty (žák a učitel) a popíše, jaké operace dělají (málo zastoupená)
- po cca 10 minutách učitel ukončí práci na algoritmech, vybere si zástupce přístupů popsaných výše
- následuje diskuse, ve které učitel vede žáky k pochopení rozdílu přístupů ke tvorbě algoritmu
- po ukončení diskuse učitel vysvětlí žákům, že analyzovaná situace je analogická s komunikací s databázovým serverem a zopakuje, že ke komunikaci s SQL serverem je zapotřebí dva objekty (napsané na tabuli – SQLConnection a SQLCommand)
- učitel pokračuje v analogii, kde se zeptá, čemu ve scénce odpovídá SQLConnection - mobilnímu telefonu, následně se diskutuje, co je třeba pro navázání („vytočení") spojení s SQL serverem (nalezené analogie píše učitel na tabuli k připraveným parametrům):
    - Data Source – odpovídá telefonnímu číslu
    - Database – odpovídá osobě, se kterou chci komunikovat (třídnímu učiteli/učitelce)
    - User ID – odpovídá jménu žáka (že se představí)
    - Password – odpovídá intonaci, zabarvení hlasu, dikci[8]
- obdobně se diskutuje SQLCommand, kde analogie na scénku je, že se žák omluví. Opět se hledají analogie k argumentům:
    - Cmd text: formulace omluvy
    - Connection: použitý mobilní telefon
- učitel ukončí inscenaci slovy, že nyní si vyzkoušíme vykonání SQL příkazu programově

Cvičení

Náročnost cvičení tohoto tématu nespočívá v množství potřebného programového kódu, ale v komplexnosti úlohy. K eliminaci nadměrného množství výjimek je použito předchozí ověření parametrů. Po zprovoznění programového kódu do něj žák záměrně vkládá chybné parametry a zjišťuje, jak se program zachová (viz kapitola 1.2).

Vytvořený programový kód bude třeba v dalších tématech, proto učitel žáky instruuje, aby si jej uložili do své složky.

[8] Nalezení této analogie bývá pro žáky komplikované.

Pomůcky:

- osobní počítač s vývojovým prostředím (Visual Studio v aktuální verzi)
- databázový server (MS SQL Server Develop edition v aktuální verzi).

Učitel zadá úlohu prostřednictvím pracovního listu, který následuje.

Komunikace s databázovým serverem

Cíl:

- vytvořit konzolovou aplikaci, která načte jednoduchá data z MS SQL serveru

Rozšíření cíle:

- zapsat data na SQL server

Namespace:

- default: SPSEJecna.{třída}.{přihlašovací jméno}.SQLBase

Třída:

- DataKontroler: třída bude zajišťovat úlohy potřebné pro komunikaci s SQL serverem

Dokumentace aplikace:

- xml automatická

Parametry připojení k databázovému serveru[9]:

- server: …
- databáze: …
- uživatel: …
- heslo: …

Úkoly:

- vytvořit konzolovou aplikaci
- nastavit defaultní namespace
- přidat nový soubor typu class pro třídu DataKontroler
- ve třídě DataKontroler vytvořit statickou metodu, která otevře spojení a vykoná sql příkaz
    - sestavit connection string
    - vytvořit připojení k SQL serveru (objekt SqlConnection)
    - otevřít spojení
    - vytvořit příkaz (objekt SqlCommand)
    - odeslat příkaz na SQL server a zpracovat výsledek

[9] Informace potřebné ke spojení s databázovým serverem. Může se jednat o lokální server, nebo síťový.

Testování:

- vytvořit spojení s SQL serverem a databází (SQLConnection)
    - vytvořit instanci objektu SQLConnection a použít parametry připojení (viz výše)
    - vyzkoušet otevření spojení
    - změnit některý z parametrů připojení za nesprávný a zjistit, jak se aplikace zachová
- provést příkaz na SQL serveru (SQLCommand)
    - vytvořit instanci objektu SQLCommand
    - vyzkoušet SQL příkaz, který:
    - vrátí počet záznamů z libovolné tabulky
    - odstranění některého řádku
    - pro spojení s databázovým serverem použít SQLConnection z předchozího úkolu
    - vyvolat SQL příkaz pomocí metody:
    - ExecuteScalar: pokud chcete vrátit jednoduchá data ze serveru
    - ExecuteNonQuery: pokud chcete příkaz pouze vyvolat a nepožadujete nic vrátit
    - změnit sql příkaz, aby nebyl validní a zjistit, jak se aplikace zachová

Ošetření výjimek:

- spojení s SQL serverem
- vykonání SQL příkazu

Kontrolní otázky:

- jaká třída zajišťuje spojení s databázovým serverem
- jaké základní argumenty jsou třeba pro navázání spojení s databázovým serverem
- jaká třída slouží k vykonání SQL příkazu na databázovém serveru
- jaké základní argumenty jsou třeba pro vykonání SQL příkazu
- co způsobí a jak reagovat na výjimku: ExecuteNonQuery requires an open and available Connection. The connection's current state is closed.

Domácí úkol:

- dokončit programovací kód se všemi náležitostmi

Odevzdání:

- složku projektu nakopírovat do …[10]

[10] Síťová složka určená pro odevzdávání práce žáka

2.2 Vykonání SQL příkazu s komplexním návratovým typem (reader/adapter)

Cvičením se žák naučí zpracovávat komplexní návratové typy, a to buď ve formě postupného načítání dat z databázového serveru (data reader), nebo ve formě lokální kopie dat/tabulky (data adapter).

Důvodem, proč se nacvičují dvě techniky, je, že se převážně používají při vývoji jiných typů aplikací. Postupné načítání dat se uplatňuje v technice strongly typed modelech (například ASP.Net Core aplikace). Lokální kopie tabulky se typicky používá ve Windows Forms aplikacích.

Specifické cíle

Žák:

- zvládne postupné načítání záznamů z tabulky databáze pomocí instance objektu DataReader
- zvládne „vytvoření kopie" dat z tabulky v databázi do lokální tabulky pomocí instancí objektů DataTable a DataAdapter
- vysvětlí důvody pro použití DataReaderu, nebo DataAdapteru
- zobrazí přenesená data, případně s nimi provede operaci (například spočtení průměru)

Opakování, prohloubení znalostí a motivace

Opakování je orientováno na upevnění znalostí z předchozího dvojhodinového bloku a zároveň bude sloužit jako „odrazový můstek" k procvičované látce.

K opakování je zvolena *inscenační* metoda výuky. Tato metoda vhodně demonstruje časovou náročnost vykonání SQL commandu, která se ve školních podmínkách jinak těžko předvede.[11]

Opakování má za cíl, aby si žák:

- zopakoval role a klíčové parametry objektů SQLConnection a SQL Command
- uvědomil technické souvislosti komunikace, především čas potřebný k přenosu dat
- navodit situaci, aby si žák uvědomil potřebu hromadného přenosu informace (doposud pracoval se skalární hodnotou)

[11] Pokud se používá k výuce vzdálený server, bývá jeho odpověď obvykle v řádu ms. Naopak vývojové prostředí na počítačích žáků pracují s odezvou s. Realita školního prostředí bývá značně odlišná od produkčních prostředí.

Inscenační prostor a pomůcky:

- lze využít libovolnou učebnu (nejsou speciální nároky na prostor)
- žák potřebuje papír a psací potřeby
- seznam (tabulku) žáků dle třídního výkazu. Musí obsahovat alespoň tři sloupce a to jmenný identifikátor (například přidělené jméno v informačním systému školy) a číselný identifikátor, jméno žáka (příklad viz Tabulka 2)[12]

Role:

- databázový server
- sql connection 1
- sql connection 2
- klient 1-n (zbytek žáků ve skupině)

Zadání:

- žáci mají zjistit o sobě informace, které jsou v seznamu.

Průběh:

- učitel vysvětlí zadání, jaké role vystupují a průběh. Následně žákům ukáže tabulku, ze které budou získávat informace. Strukturu tabulky je třeba napsat na tabuli, nebo promítnout.[13]
- učitel přidělí některému žákovi roli serveru a následně se jej zeptá, jaké má jméno a v jaké databázi je uložena tabulka (argument Data Source a Database při vytváření spojení). Získané informace zapíše na tabuli.[14]
- učitel přidělí dvěma žákům role sql connection. Poté je vyzve, aby sestavili connection string. Oni buď samostatně, nebo s pomocí třídy a učitele převezmou informace z databázového serveru napsané na tabuli. Problém jsou autentizační údaje - ty si musejí „dohodnout" s databázovým serverem.
- učitel dá žáku představujícímu databázový server papír s vytištěným seznamem (tabulkou) a pošle jej na chodbu, kde bude čekat na druhého žáka představujícího sql command (například na lavičce).
- učitel vyzve zbylé žáky, aby na papír napsali sql command, který vrátí jednu z informací (například SELECT id FROM tabulka WHERE user_name='klimes2').

[12] Může se jednat o libovolný seznam, který je dostupný a pro žáky srozumitelný. Seznam je možné též doplnit o nějaké atraktivní informace (například učitelovu poznámku k osobě).

[13] Lépe je ukázat seznam jiné třídy, aby žáci nepřišli o moment překvapení

[14] Zajišťuje opakování klíčových parametrů databázového serveru.

- pokud má žák sql command napsaný, pak jej předá jednomu z sql connection, který „zajde“ za databázovým serverem, který dotaz zpracuje a sql connection se vrátí s odpovědí.
- žáci opakují sql commandy a „sbírají“ informace ze seznamu.
- během inscenace mohou nastávat situace kdy:
    - více žáků musí čekat na návrat sql connection, aby jej mohli použít pro svůj sql commnad. Učitel v tento moment může na tuto situaci poukázat jako na realitu praxe a vyzvat je k návrhu řešení (například vytvořit více sql connection)
    - žák, který je klientem (vytváří sql command), se zvedne s úmyslem zajít za databázovým serverem sám. Učitel využije této situace k vysvětlení, že nemůže jít sám, ale musí použít některé z sql connection.
    - žák na papír napíše nejen sql příkaz, ale i connection string. Učitel této situace využije k zopakování, že connection string je argumentem sql connection a ne sql command.
- ve vhodnou chvíli (kdy bude mít pocit, že pro žáky to začíná být úmorné) se učitel žáků zeptá, zda by úlohu nešlo nějak urychlit. Snaží se je navést na odpověď, že by šlo poslat všechny data najednou.
- učitel se zeptá, jak by vypadal sql příkaz, který pošle všechna data najednou (například SELECT * FROM tabulka).
- příkaz předá jednomu z SQL connection a řekne mu, aby převzal papír se seznamem a přivedl databázový server nazpět.
- učitel převezme od sql connection papír s tabulkou a řekne žákům, že nyní máme k dispozici všechna data, ale jak si ji můžeme přečíst? Možnosti jsou:
    - řádek po řádku. Učitel může demonstrovat tak, že papír s tabulkou dá prvnímu žákovi a vyzve jej, aby přečetl první řádek a předal dalšímu žákovi, který přečte druhý řádek (stačí cca 4-5 řádků). Učitel vysvětlí žákům, že tato metoda odpovídá použití instance objektu DataReader.
    - vytvořit si lokální kopii. Učitel může dát papír s tabulkou některému žákovi a ptát se jej na různé otázky ohledně dat. Učitel vysvětlí žákům, že tato metoda odpovídá po použití DataTable (pro uložení lokální kopie) a DataAdapter, pro vytvoření lokální kopie
- učitel ukončí inscenaci tím, že nyní si to vyzkoušíme DataReader, DataAdapter a DataTable programově

Tabulka 2: Výkaz žáků (inscenace komunikace s databázovým serverem)

C2b výkaz				
Id	name[15]	Group	user_name	specialization
1	B… Milan	A1, O	b…	C
2	B… Vojtěch	A1, O	b…	C
3	D… Andriy	A1, O	d…	C
4	D… Jan	A1, O	d…	C
5	H… Robin	A1, O	h…	C
6	Ch… Jiří	A1, O	ch…	C
7	J… Arman	A1, O	j…	C
8	J… Petr	A1, O	j…	C
9	J… Thang	A1, O	j…	C
10	K… Michal	A1, O	k…	C
11	K… Tomáš	A1, O	k…	C
12	K… Šimon	A1, O	k…	C
13	K… Martin	A1, O	k…	C
14	K… Martin	A1, O	k…	C
15	K… David	A2, O	k…	C
16	K… Svetlana	A2, O	k…	C
17	K… David	A2, O	k…	C
18	M… Lukáš	A2, O	m…	C
19	P… Radim	A2, O	p…	C
20	P… Ondřej	A2, O	p…	C
21	P… Jiří	A2, O	p…	C
22	P… Jiří	A2, O	p…	C
23	R… Milan	A2, O	r…	C
24	Š… Roman	A2, O	s…	C
25	Š… Hugo	A2, O	s…	C
26	Š… Jaroslav	A2, O	s…	C

Zdroj: vlastní zpracování

[15] Příjmení a uživatelská jména byla zkrácena z důvodu ochrany soukromí žáků.

Cvičení

Obtížnost tohoto cvičení pro žáka spočívá v nácviku dvou výrazně odlišných technik pro jednu úlohu: získat větší množství dat data z SQL serveru. Rizikem je, že žák může přestat rozlišovat, kterou techniku právě nacvičuje. Učitel tomu může předcházet tím, že zdůrazňuje, která technika se právě nacvičuje.

Pomůcky:

- osobní počítač s vývojovým prostředím (Visual Studio v aktuální verzi)
- databázový server (MS SQL Server Develop edition v aktuální verzi).
- databázi s tabulkou, kde jsou uložena cvičná data
- zajištěný přístup na SQL server/databáze/tabulky

Učitel zadá úlohu prostřednictvím pracovního listu, který následuje.

Získání komplexních dat z databázového serveru

Cíl:

- rozšířit konzolovou aplikaci z přechozí hodiny o možnost načtení komplexních dat

Rozšíření cíle:

- zapsat data na SQL server

Úkoly:

- ve třídě DataKontroler založit statickou metodu, která vytvoří instanci třídy DataReader
- ve třídě DataKontroler založit statickou metodu, která vytvoří instanci třídy DataAdapter

Testování:

- vytvořit instanci DataReader
  - vrátit hodnotu pomocí indexu pole v určitém datovém typu (GetInt32, GetString, ...)
  - vrátit hodnotu pomocí jmenného názvu (použití pole)
  - ošetřit hodnotu DBNull
- vytvořit instanci DataAdapter a DataTable
  - načtená data strukturovaně vypsat

Ošetření výjimek:

- vyvolání DataReader
- nastavení DataAdapter

Kontrolní otázky:

- k čemu slouží třída DataReader
- k čemu slouží třída DataAdapter
- jaký je rozdíl při použití DataReader a DataAdapter

Odevzdání:

- složku projektu nakopírovat do …[16]

[16] Síťová složka určená pro odevzdávání práce žáka

3 Tvorba grafického uživatelského rozhraní (GUI)

Aplikace s konzolovým rozhraním mají své pevné místo v oblasti správy informačních systémů. „Běžný“ uživatel však očekává, že aplikace bude mít grafické uživatelské rozhraní (GUI). Tvorby GUI je proto důležitou kompetencí žáka.

Žák během studia využívá pro většinu úloh rozhraní konzole. Důvody lze spatřovat v:

- *nízké časové i mentální náročnosti tvorby rozhraní*: žák se tak může plně soustředit na probírané téma
- *jednoduše organizovaném kódu*: v konzolové aplikaci lze potřebný kód napsat do startovací metody Main(). Odpadá tak náročnější organizace programového kódu v případě grafického prostředí.
- *pevném pořadí uživatelských vstupů*: konzolová aplikace ze své podstaty umožnuje pouze jeden vstup a žák nemusí implementovat kontrolu logických vazeb. Například, pokud žák v úloze nejprve potřebuje znát počet prvků a pak jejich hodnoty, pak tomu uzpůsobí sled dotazů a načtení hodnot. Uživatel pak nemůže postupovat v jiném pořadí a tím narušit fungování programu.

Tvorba grafického rozhraní přináší žákovi řadu nových podmětů. Žák tak má možnost rozvíjet odlišné kompetence, než které jsou potřeba pro kódování samotné funkcionality.[17]

Vliv velikosti zobrazovací plochy

Velikost zobrazovací plochy cílového zobrazení je pro tvorbu GUI klíčová. Grafická uživatelská rozhraní lze podle cílového „zařízení“ rozdělit na:

- desktopová: určená pro osobní počítače, obvykle s velkou zobrazovací plochou
- webová: určená pro webové prohlížeče, s nutností pracovat s velkou i malou zobrazovací plochou
- mobilní: určená pro mobilní zařízení, většinou s malou zobrazovací plochou

Každá varianta GUI má své principy a pravidla návrhu, které nelze zaměňovat[18]. Optimálně je vhodné do výuky zařadit všechny tři varianty.

[17] Nezřídka se stává, že žák, který je „zručný“ kodér, má problém vytvořit uživatelsky přívětivé a estetické grafické rozhraní. Naopak, mohou vyniknout žáci, kterým se kódování příliš nedaří.

Problematika výuky

Problematiku tvorby grafického uživatelského rozhraní lze shrnout:

- žáci mají v oblibě toto téma[19]
- výsledky úloh jsou „atraktivnější“: žák může sdílet svou práci například s rodiči, kamarády a tím si vytvářet pozitivní vazbu na předmět

- náročné na tvorbu: vytvoření i jednoduchého funkčního rozhraní je časově i mentálně náročné.
- GUI je zdrojem specifických výjimek: například desktopové grafické rozhraní je „postavené“ na implementaci objektových událostí. Odstranění nechtěné události je třeba provést na dvou odlišných místech (ve dvou programových souborech), jinak aplikace hlásí výjimku.

Přínos zvládnutí tohoto tématu lze spatřovat v:

- žák si uvědomí význam i náročnost tvorby GUI
- žák rozvíjí své graficko-organizační schopnosti a estetické vnímání (tj. nejen schopnost programovat)

Technologie a prostředí

Pro cvičení tématu bylo zvoleno:

- uživatelské prostředí Visual Studio (v aktuální verzi)
- programovací jazyk C#
- vývojová platforma Windows Forms

Koncipování výuky

Téma je natolik rozsáhlé, že je vhodné jej rozdělit na obecné uvedení do problematiky a specifičtější dovednosti „zamíchat“ do dalších témat. Obecné uvedení do problematiky se skládá ze dvou částí (dva dvojhodinové bloky), první se věnuje práci s formulářem a kontrolem, druhý se věnuje provazování dat a vlastností kontrolů.

Náročnost úlohy

Náročnost tématu vychází z nutnosti volby z více konkurenčních platforem. Například jen Microsoft aktuálně nabízí pro vývoj desktopových aplikací čtyři platformy (Desktop, 2019). Pokud se vynechají platformy UWP a Win32Api (které teprve nastupují, nebo jsou určené pro C/C++), pak zbydou dvě „zavedené“.

---

[18] Například v desktopových aplikacích často bývá v dolní části informační řádek, například u Microsoft Word s číslem řádky, na které se uživatel nachází, počtu slov, jazyku a podobně. Ve webovém GUI musejí být stejné informace naopak v horní části, neboť webová stránka je zdola neohraničená.

[19] Zřejmě je pro žáky práce na grafickém rozhraní naplnění představy, co se budou na škole zaměřené na programování učit. Jistým rozčarování pro ně může být, že se k programování GUI dostávají obvykle až ve třetím ročníku.

Windows Presentation Foundation (WPF) (WPF, 2018)

- ✓ novější/modernější rozhraní
- ✓ možno demonstrovat návrhový vzor MVVM (MVVM, 2009)
- ✗ náročnější na pochopení
- ✗ „nepřehledná" pro začínající vývojáře

Windows Forms .NET (WinForms) (WinForms, 2017)

- ✓ intuitivní
- ✓ silně zastoupená v podnikových aplikacích
- ✗ není „moderní" (starší než WPF)

Volba platformy je daná učitelem, respektive školním vzdělávacím plánem, kde možné preference shrnuje Tabulka 3.

Tabulka 3: Porovnání platformy Windows Forms a WPF z hlediska preferencí

Preference	*Windows Forms*	*WPF*
zkušenost s grafickým rozhraním	✓ intuitivní	✗ méně přehledné
dovednosti v konkrétní platformě	✗ „zastaralá"	✓ „moderní"
uplatnění v praxi	✓ především podnikové aplikace, které se udržují a rozvíjí	✓ nově vyvíjené aplikace
demonstrace návrhového modelu	✗ není specifický návrhový model	✓ MVVM

Zdroj: vlastní zpracování

3.1 Návrh formuláře, vkládání kontrolů a užití událostí

Cvičení zprostředkuje žákovi dovednosti potřebné pro návrh grafického prostředí aplikace a jeho zprovoznění. Vede jej k uvědomění potřeby nejen kódovat, ale také vytvářet grafické návrhy a rozvoje estetického vnímání.

Specifické cíle

Žák:

- porozumí návrhu grafického rozhraní založeného na událostech
- rozumí významu formuláře pro návrh grafického rozhraní
- vhodně volí kontroly a umísťuje je na formulář
- nastavuje potřebné vlastnosti formuláře a kontrolů
- vytváří a odstraňuje potřebné události formuláře a kontrolů
- uvědomuje si vztahu mezi nástrojem designer a programovým kódem

Opakování, prohloubení znalostí a motivace

Ke tvorbě grafického rozhraní bude žák potřebovat „přetvořit" zadání do kooperativní spolupráce grafických prvků. K jeho „oživení" zase rutinní práci s objektovými událostmi.

Návrh grafického rozhraní pro desktopové aplikace je úzce spjat se zachytáváním objektových událostí kontrolů. Objektové události mohou být samostatně probrány a procvičeny v předchozích tématech (obvykle společně se základy objektového programování). V tomto tématu se žák k objektovým událostem vrací ne jako k předmětu studia, ale jako k nezbytnému prostředku tvorby grafického rozhraní.

K uvědomění si vazby mezi grafickým prvkem a požadovanou funkcionalitou byla zvolena heuristická metoda.

Opakování má za cíl, aby si žák:

- uvědomil vazbu mezi grafickým prvkem a jeho role v požadované funkcionalitě
- uvědomil, že aplikace funguje, když se například „klikne" na grafický prvek

Prostor a pomůcky:

- běžná učebna, kde budou mít žáci dost místa na stole[20]
- žák potřebuje papír, psací potřeby
- stavební kostky z dětské stavebnice. Je vhodné zahrnout malé a střední kostky různých barev (budou reprezentovat kontroly), velké desky (budou reprezentovat formulář)[21]

Zadání:

- žáci mají pomocí stavebních kostek navrhnout a sestavit jednoduchou kalkulačku. Dále mají napsat na papír, jak bude fungovat v návaznosti na „stavební kostky"[22] (ukázka výstupů viz Obrázek 6 a Zdroj: studentská práce
- Obrázek 7).

Průběh:

- učitel vysvětlí zadání úlohy, během kterého vyndá na volnou lavici stavební kostky

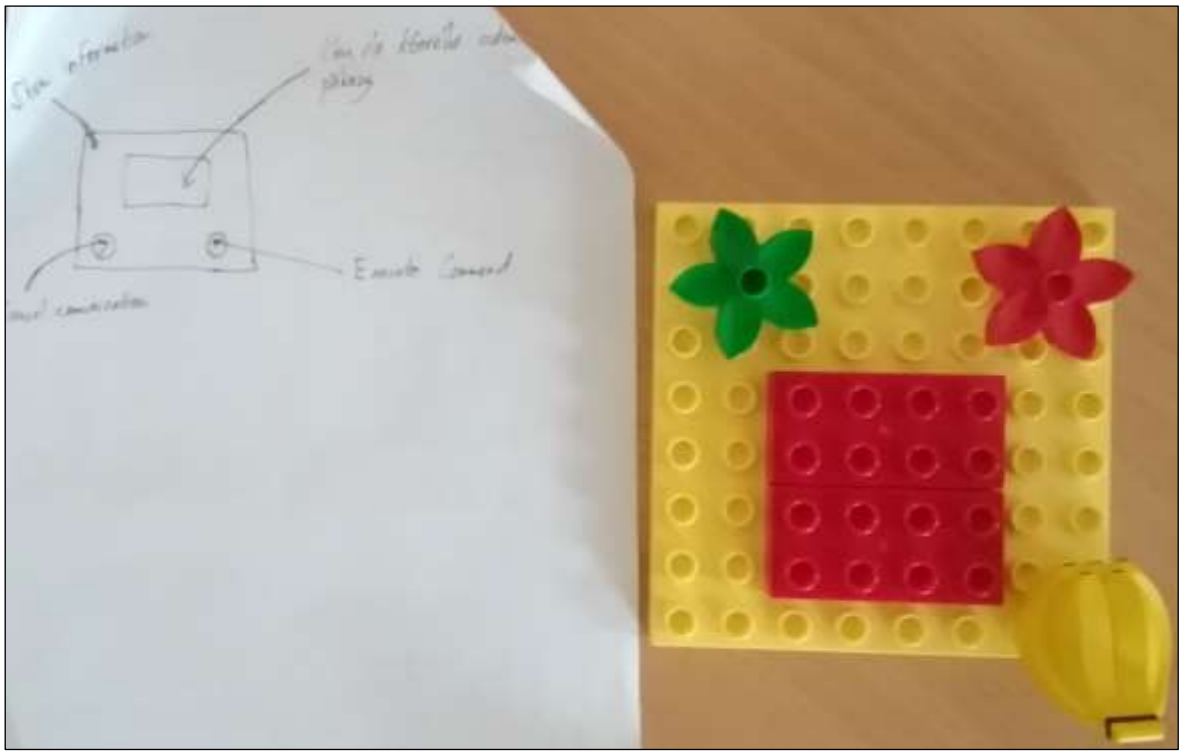
[20] Odborná učebna bývá nevyhovující (stůl je zaplněn počítačovými periferiemi), je vhodné si předem vyhledat a zajistit vhodnou učebnu

[21] Zlepšení nálady mezi žáky přispěje, pokud se přidají kostky jako banán, kytička a podobně. Tyto kostky někteří žáci vhodně použijí jako třeba ovládací prvky.

[22] Pro některé žáky může být tato úloha náročná (například u žáků s poruchou autistického spektra), úlohu lze pro ně modifikovat, že grafický návrh provedou na papír.

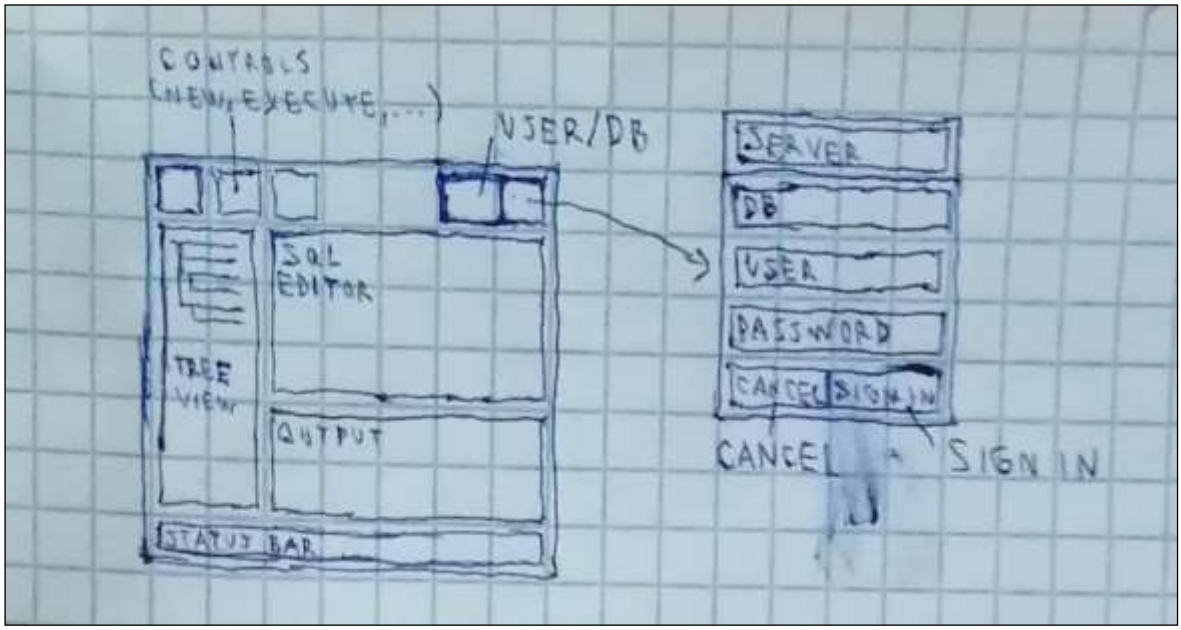
- učitel vyzve žáky, aby vytvořili dvojice až trojice (v závislosti na počtu žáků a množství stavebních kostek)
- učitel vyzve skupinky, aby si přišly vybrat stavební kostky, které budou potřebovat a začaly na úloze pracovat
- po cca pěti minutách (které lze využít pro zápis do třídní knihy) učitel obchází skupiny a nechá si vysvětlit, co ta která kostka reprezentuje
- žáci mají často problémy s písemnou formulací funkcionality. Pro názorné vysvětlení lze použít techniku, kdy učitel si nejprve nechá od žáků vysvětlit, jak jejich kalkulačka funguje a poté jim řekne:„...a teď stačí stručně napsat, co jsi mi řekl."
- postupně učitel obejde všechny skupinky a nechá si popsat jejich řešení a porovná jej s „grafickým" návrhem a popisem funkcionality na papíře
- učitel vybere vhodnou práci žáků, kterou použije pro demonstraci:
    - zeptá se, jaký je ekvivalent základové desky v programu, tj. toho, co „drží" všechno pohromadě - formulář (objekt Form)
    - zeptá se, jaký je ekvivalent kostek vsazených do základové, tj. toho, co něco dělá, k něčemu slouží – kontrol (objekt Control)
    - zeptá se, jaký je ekvivalent když něco „zmáčknu", tj. když potřebuji něco udělat – objektová událost (Event)
- učitel ukončí úlohu slovy, že nyní si vytvoříme kalkulačku v počítači

Obrázek 6: Návrh grafického rozhraní žákem s použitím stavebních kostek a popisu



Zdroj: studentská práce

Obrázek 7: Návrh grafického rozhraní žákem pouze na papír (bez použití stavebních kostek – porucha autistického spektra)



Zdroj: studentská práce

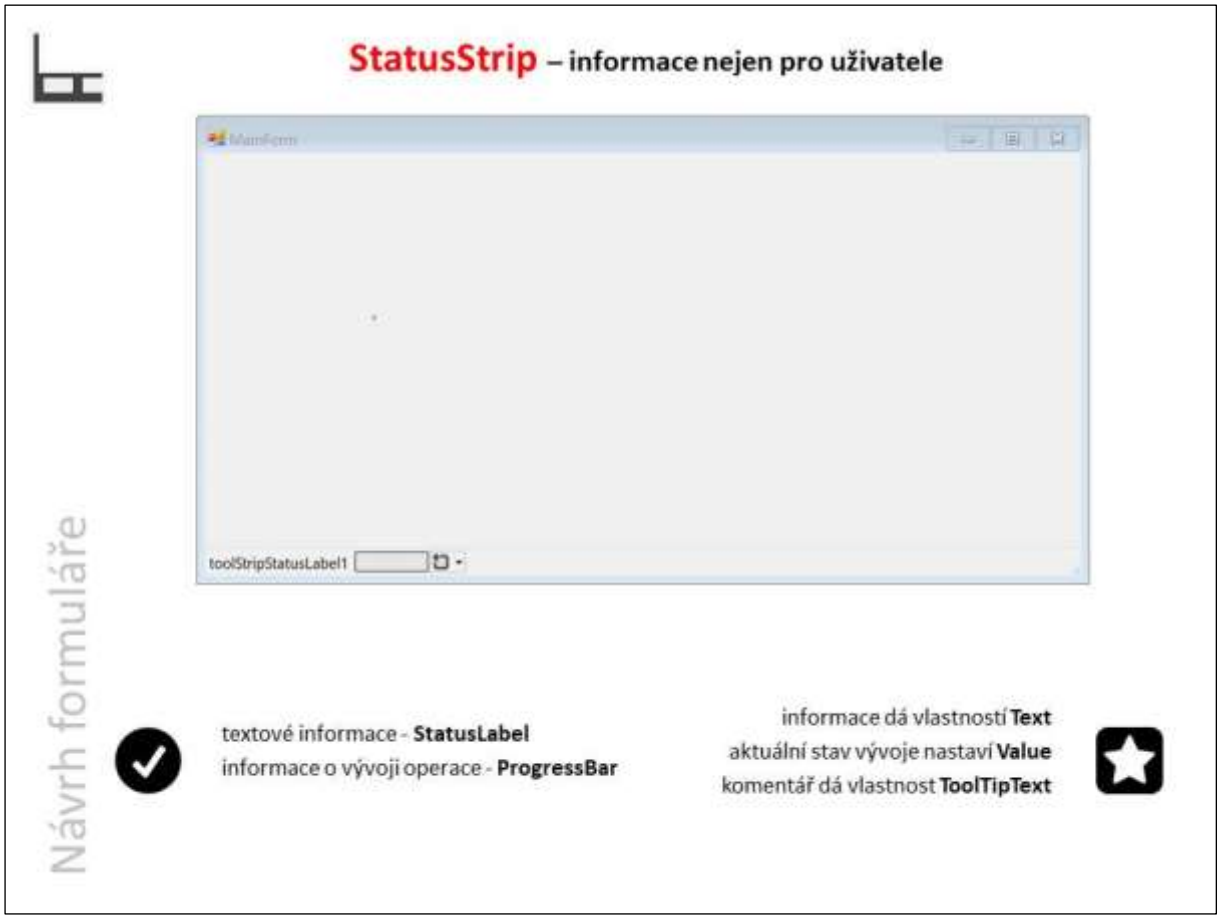
Cvičení

Cvičení zahrnuje nejen samotnou tvorbu grafického rozhraní, ale také způsob, jak se v tomto případě pracuje s vývojovým prostředím. Žáka je třeba seznámit:

- jaký typ projektu je vhodný
- jak se zobrazí panel s kontroly (Toolbox)
- jak se přepíná mezi designem a programovým kódem
- jak se generuje/odstraní událost ke kontrolu

Žák při návrhu formuláře má k dispozici velké množství kontrolů,.jen systémových je v .Net Frameworku 50 (Controls, 2017). Učitel může žákovi doporučit kontroly, které jsou vhodné pro řešení zadané úlohy. Vhodným prostředkem je vytvořit „průvodce" běžnými kontroly, kde je vysvětleno, k čemu se kontrol používá a jaké jsou jeho základní vlastnosti. Příkladem může být snímek z průvodce ke kontrolu StatusStrip (viz Obrázek 8).

Obrázek 8: Snímek z průvodce kontroly



Zdroj: vlastní zpracování

Pomůcky:

- osobní počítač s vývojovým prostředím (například Visual Studio v aktuální verzi)

Pracovní list:

Kalkulačka [WinForms]

Cíl:

- vytvořit WinForms aplikaci, kalkulačku, která zvládá operace sčítání, odčítání, násobení a dělení

Rozšíření cíle:

- doplnit další operace dle vlastní volby
- zobrazovat historii výpočtů
- umožnit ukládání výsledků do souboru

Namespace:

- default: SPSEJecna.{třída}.{login}.BasicCalculator

Interface/Abstract class:

- IBasicCalculator

Konfigurace:

- není

Dokumentace:

- xml automatická

Zadání úkolu:

- vytvořit WinForms aplikaci
- nastavit defaultní namespace
- pomocí kontrolů navrhnout vzhled formuláře
- vytvořit interface a k ní třídu, která bude zajišťovat matematické operace
- propojit formulář s třídou prostřednictvím vhodných událostí

Ošetření výjimek:

- vstup uživatele není číslo
- přetečení rozsahu čísel (number overflow)

Kontrolní otázky:

- co je charakteristické pro WinForms aplikaci
- jaké úlohy jsou typické pro WinForms aplikaci
- jak se aplikaci předává činnost uživatele

Odevzdání:

- složku projektu nakopírovat do …

3.2 Propojení vlastnosti kontrolu s daty

Desktopové aplikace jsou často zacíleny pro práci s daty. Vývojová prostředí proto obsahují efektivní způsob, jak s daty v aplikaci pracovat. Cvičení má za cíl, aby si žák osvojil způsob provazování dat s kontroly. Dále cvičení umožňuje syntézu dříve probíraných témat s aktuálním tématem – komunikace s databází a grafické rozhraní.

Specifické cíle

Žák:

- zvládne propojit vlastnost kontrolu s datovým prvkem
- dokáže zobrazit hromadná data pomocí datové tabulky (DataGridView)
- aktivně používá objekty k procházení dat

Opakování, prohloubení znalostí a motivace

Zdárné řešení úlohy vyžaduje syntézu znalostí a dovedností z několika předchozích cvičení (SQL komunikace, vytvoření lokální kopie dat, návrh a zprovoznění formuláře). Blokové opakování by vyžadovalo neúměrně času a žák by byl zřejmě zahlcen.

Možným způsobem je integrovat opakování do samotného cvičení, například ve formě mikro výkladu - dílčí krok. Před samotným cvičením se probere rámcové řešení úlohy. Vhodnou metodou je problémová úloha.

Opakování má za cíl, aby si žák:

- uvědomil „aktéry" úlohy
- rozpoznal důvody, proč se přistupuje k danému řešení
- uvědomil si princip vazby mezi daty a kontroly

Prostor a pomůcky:

- odborná učebna
- školní tabule nebo projektor
- žák potřebuje papír, psací potřeby

Zadání:

- jak zobrazit data ve formuláři

Průběh:

- učitel nejprve nakreslí na školní tabuli/zobrazí na projektoru „aktéry" úlohy
- učitel položí otázku, jak pomocí kontrolu zobrazím například text
    - odpověď: použiji kontrol TextBox a nastavím vlastnost Text
- učitel nakreslí na školní tabuli/zobrazí na projektoru TextBox a napíše vlastnost Text

- učitel pokračuje otázkou, jak zobrazit data uložená v lokální tabulce (objektu DataTable)
    - odpověď: nastaví se vlastnost Text kontrolu TextBox hodnotou z nějakého řádku a sloupce z DataTable
- učitel nakreslí na školní tabuli/zobrazí na projektoru Tabulku, která reprezentuje DataTable
- učitel otočí úlohy a zeptá se, jak lze z kontrolu TextBox přepsat hodnotu do tabulky DataTable
    - odpověď: pomocí události TextChanged
- učitel nastíní problém:
    - pokud budeme potřebovat provázat velké množství dat, bude třeba velké množství programového kódu
    - co když je třeba jednu hodnotu měnit z více míst
- učitel řízenou komunikací se žáky směřuje k tomu, že řešením je vytvořit vazbu, která se bude „sama" starat o předávání hodnot mezi kontrolem a datovou tabulkou
- učitel nakreslí na školní tabuli/zobrazí na projektoru vazbu mezi kontrolem a tabulkou. K vazbě připíše slovo Bind, které žákům přeloží do češtiny.
- učitel ukončí opakování slovy: „Nyní si to vyzkoušíme prakticky."

Cvičení

- Cvičení využívá znalostí a programového kódu z předchozích témat. Pokud žák nemá k dispozici vlastní kód z předchozích hodin, pak mu jej může poskytnout spolužák, nebo učitel.
- Úloha je natolik komplexní, že ji učitel může vhodně rozložit do jednotlivých kroků. Po každém kroku učitel zkontroluje, že jej žáci zvládli. Před každým krokem může učitel zopakovat, jaký je jeho účel a co k němu bude potřeba.

Pomůcky:

- osobní počítač s vývojovým prostředím (Visual Studio v aktuální verzi)
- databázový server (MS SQL Server Develop edition v aktuální verzi).
- databázi s tabulkou, kde jsou uložena cvičná data
- zajištěný přístup na SQL server/databáze/tabulky

Pracovní list:

SqlTableViewer [WinForms]

Cíl:

- vytvořit WinForms aplikaci, která umožní prohlížet tabulku z databázového serveru

Rozšíření cíle:

- umožnit aktualizovat údaje a ukládat je zpět na databázový server

Namespace:

- default: SPSEJecna.{třída}.{login}.SqlTableViewer

Interface/Abstract class:

- není

Konfigurace:

- připojovací řetězce k databázi (connection string)

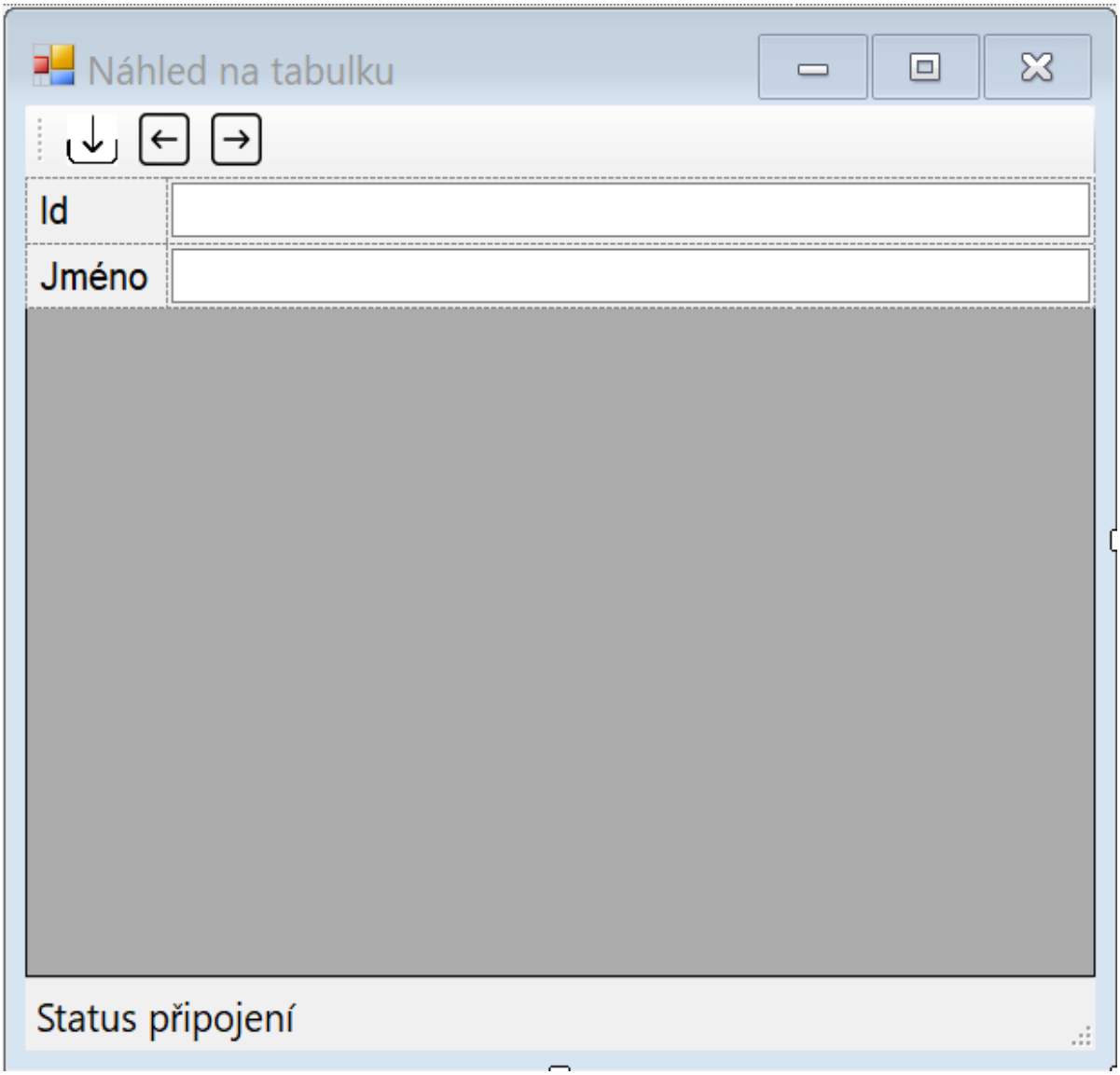
Dokumentace:

- xml automatická

Zadání úkolu:

- vytvoř WinForms aplikaci
- nastav defaultní namespace
- navrhni formulář dle vzoru (viz níže). Formulář obsahuje:
    - ToolStip + 3 x Button
    - TableLayoutPanel
    - 2 x Label
    - 2 x TextBox
    - DataGridView
    - StatusStrip
- ikony k tlačítkům jsou uložena v…
- použij kód pro načtení lokální tabulky z databázového serveru
- vytvoř objekt BindingSource, který zajišťuje ukazatel na aktuální řádek v DataTable
- vytvoř propojení mezi kontroly TextBox a polem v tabulce – objekt Binding
- nastav vlastnost BindingSource u DataGridView

- ověř funkčnost kódu



Ošetření výjimek:

- komunikace s SQL serverem
- provázání DataTable s vlastností kontrolu

Kontrolní otázky:

- pomocí jakých informací se vytváří vazba (objekt Bind)
- jaký je přínos objektu BindingSource

Odevzdání:

- složku projektu nakopírovat do …

4 Klasifikační zkouška

Klasifikační zkouška pro rozpracovaná témata je vedena ve formě písemné práce. Nejsou požadovány přesné (normalizované) formulace odpovědí. Žák je spíše nabádán k vlastnímu vyjádření, ze kterého je možné diagnostikovat míru pochopení tématu.

Jedná se o kriteriální zkoušku, která je však v případě výrazného nezdaru normována. Důvodem je přítomná „nejistota" v nastavení výukových cílů, zvolených forem a podobně. Nejedná se o test výkonu, proto žáci mají k dispozici až dvě vyučovací hodiny (celý blok)[23]. Zbylý čas (pokud nastane) lze využít k probrání možných odpovědí a k diskusi.

Zadání práce je formou pracovního listu), do kterého žák doplňuje své odpovědi (viz Obrázek 9 a Obrázek 10).

[23] Zkušenost autora ukazuje, že obvykle žáci potřebují jednu vyučovací hodinu.

.Obrázek 9: Pracovní list klasifikační zkoušky tématu „Komunikace s databázovým serverem“

C# Komunikace s DB	Jméno a přijmení:		Třída:	
Sekce	Úloha		Odpověď	Body
Spojení s DB serverem	Název objektu			1
	Popis funkcionality			1
	Klíčové argumenty pro vytvoření spojení:	1.		1
		2.		1
		3.		1
Vykonání SQL scriptů	Název objektu			1
	Popis funkcionality			1
	Klíčové argumenty potřebné pro vykonání	1.		1
		2.		1
SQLDataAdapter	Popis funkcionality			1
	Klíčové argumenty potřebné pro vytvoření	1.		1
		2.		1
		3.		1
		4.		1
SQLDataReader	Popis funkcionality			1
	Klíčové argumenty potřebné pro vytvoření	1.		1
		2.		1
DataTable	Popis funkcionality			2
Sestav dvojice, které spolu **přímo** souvisejí a popiš jejich vazbu	User ID DataDatapter SQLDataReader SQLConnection Fill Select command SQLCommand ExecuteReader sqlScript Database DataTable	1.		1
		2.		1
		3.		1
		4.		1
		5.		1
		6.		1

Zdroj: vlastní zpracování

Obrázek 10: Pracovní list klasifikační zkoušky tématu „Windows Forms“

C# Win Forms	Jméno a příjmení:		Třída:	
Sekce	Úloha		Odpověď	Body
Návrh formuláře	K čemu slouží formulář (objekt Form)			1
	K čemu slouží kontroly (objekt děděný od Control)			1
	Napíš název kontrolu (přibližný) a popiš, k čemu slouží	1.		1
		2.		1
		3.		1
BindingSource	Co objekt zajišťuje			1
	Popis funkcionality			1
	Klíčové argumenty potřebné pro vykonání	1.		1
		2.		1
New Binding("BackColor", data, "color")	Popis funkcionality			1
	popiš argumety	1.		1
		2.		1
		3.		1

Zdroj: vlastní zpracování

Závěr

Téma bakalářské práce vycházelo z potřeby disponovat prostředky k efektivní výuce programování. Prvotní představou byla tvorba pracovních listů. Tato představa se záhy rozšířila na celkové řešení výuky vybraných témat, od volby cílů, výukových forem, metod po jejich pedagogickou diagnostiku. Prvotní ověření návrhu ukazovalo, že vsazení pracovních listů do celkové koncepce výuky je správnou cestou.

Prvotní ověřování též ukázalo důležitost volby cílů, především naplnění principu přiměřenosti. Svědomitě připravená výuka selhávala, pokud bylo množství probírané látky neúměrné potenciálu žáka. Získané zkušenosti by tak bylo vhodné transformovat do tematických plánů.

V procesu výuky se nepředávají jen znalosti, ale také postoj k předmětu, který je pro další rozvoj kompetencí žáka neméně důležitý. Podat vyučovanou látku zajímavě je náročný úkol a k jeho řešení byly zvoleny formy výuky, které kladou důraz na názornost, tvořivost a aktivní komunikaci žáka s učitelem.

Práce neobsahuje problematiku domácích úkolů. Tuto problematiku se nepodařilo uspokojivě řešit. Důvodem byla především neúměrně vysoká náročnost potřebná k jejich opravování a podávání zpětné vazby žákovi. Tvorba vhodných zadání domácích úkolů, spolu s formou jejich odevzdání a opravy, by mohla být tématem pro pokračování této práce.

Oblast diagnostiky je součástí práce v základní rovině. Vhodným pokračováním by bylo vytvořit nestandardizované testy, které by lépe odrážely znalosti žáků. Významným úkolem by též bylo zvládnout zkoušení v podobě praktické práce, která je ideálem pro tento předmět.

Znalosti žáků jsou přiměřeně měřitelné, zato jejich postoje lze jen těžko indikovat - snad pocitem, jaký si učitel z hodin odnáší. Dobrý pocit, který si učitel z hodin odnáší je nejen zpětnou vazbou, ale také „palivem“, bez kterého bude učitel jen těžko hledat nové cesty ve výuce svého předmětu. Dobře promyšlená a připravená výuka je základem úspěšné výuky a při troše štěstí učiteli dobrý pocit přinese – k čemuž měla tato práce přispět.

Seznam použité literatury

Algoritmus. (nedatováno). *Algoritmus.* Načteno z Wikipedie: https://cs.wikipedia.org/wiki/Algoritmus

Controls, W. (30. 03 2017). *Windows Forms Controls by Function*. Načteno z Microsoft .Net: https://docs.microsoft.com/en-us/dotnet/framework/winforms/controls/windows-forms-controls-by-function

Desktop, W. (4. 11 2019). *Choose your app platform*. Načteno z Windows Dev Center: https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform

Hrabal, V., & Pavelková, I. (2010). *Jaký jsem učitel.* Portál.

Microsoft. (2020). *Microsoft .Net*. Načteno z HttpClient Class: https://docs.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?f1url=https%3A%2F%2Fmsdn.microsoft.com%2Fquery%2Fdev16.query%3FappId%3DDev16IDEF1%26l%3DEN-US%26k%3Dk(System.Net.Http.HttpClient);k(DevLang-csharp)%26rd%3Dtrue&view=netframework-4.8

Ministerstvo školství, m. a. (29. 5 2008). RVP 18 – 20 – M/01 Informační technologie. Načteno z http://zpd.nuov.cz/RVP/ML/RVP%201820M01%20Informacni%20technologie.pdf

MVVM, M. (1. 2 2009). Načteno z https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

NPIČR, N. p. (2020). *Informační systém o uplatnění absolventů škol na trhu práce*. Načteno z infoabsolvent: https://www.infoabsolvent.cz

Pattern. (nedatováno). *Návrhový vzor.* Načteno z Wikipedie: https://cs.wikipedia.org/wiki/Návrhový_vzor

Silverlight, M. (2020). *Silverlight*. Načteno z https://www.microsoft.com/silverlight/

Vaněček, D. (2016). *Didaktika technických odborných předmětů.* Praha: ČVUT Praha.

VS-intellisense. (9. 3 2020). *IntelliSense*. Načteno z https://code.visualstudio.com/docs/editor/intellisense

WinForms. (30. 3 2017). *Windows Forms*. Načteno z Microsoft DotNet: https://docs.microsoft.com/cs-cz/dotnet/framework/winforms/

WPF. (25. 1 2018). *Windows Presentation Foundation*. Načteno z Microsoft DotNet: https://docs.microsoft.com/cs-cz/dotnet/framework/WPF/

Seznam obrázků

Seznam tabulek

Evidence výpůjček

Prohlášení:

Dávám svolení k půjčování této bakalářské práce. Uživatel potvrzuje svým podpisem, že bude tuto práci řádně citovat v seznamu použité literatury.

Jméno a příjmení: Ondřej Tomáš Klimeš

V Říčanech dne: 05. 05. 2020 Podpis:

Jméno	**Oddělení/ Pracoviště**	**Datum**	**Podpis**