STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12: tvorba učebních pomůcek, didaktická technologie

Název práce:

Videokurz a sbírka úloh pro předmět programování

Vladimír Vávra
Hlavní město Praha

Praha 2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12: tvorba učebních pomůcek, didaktická technologie

Název práce:
Videokurz a sbírka úloh pro předmět programování

Title:
Video course and collection for the subject of programming

Autoři: Vladimír Vávra
Škola: Arabská 14, Praha 6, 160 00
Kraj: Hlavní město Praha
Konzultant: Ing. Daniel Kahoun

Praha 2021

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této 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) ve znění pozdějších předpisů.

V …… dne ……. Podpis: …

Poděkování

Rád bych na tomto místě poděkoval svému učiteli programování, ing. Danielu Kahounovi, který mi vnukl nápad na tuto práci a po celou dobu ročníkové práce odpovídal na mé dotazy ohledně práce.

Anotace

Tato práce má za cíl vytvořit ucelený výukový systém, který zjednoduší výuku programování pro začínající studenty programování. Hlavní částí práce je videokurz, který kopíruje a částečně rozšiřuje učební plán 1. ročníku programování na Gymnáziu Arabská. Díky němu získají studenti základní znalosti programování, s jejichž znalostí jsou schopni již samostatně vyhledávat další potřebné informace. Ke každému videu jsou dostupné testové otázky, včetně praktických úloh na procvičení dané látky. Všechny tyto úlohy jsou automaticky oznámkované a student okamžitě dostane zpětnou vazbu. Práce také zkoumá, jak je možné tyto výukové materiály začlenit do každodenní výuky, ať už prezenční či distanční.

Klíčová slova

videokurz, Java, úlohy, začlenění do výuky, Moodle

Annotation

This work aims to create a comprehensive learning system that simplifies the learning of programming for beginning students of programming. The main part of the work is a video course, which copies and partially expands the curriculum of the 1st year of programming at Gymnázium Arabská. Thanks to it, students will acquire basic knowledge of programming, with the knowledge of which they are able to independently search for other necessary information. Test questions are available for each video, including practical exercises. All of these assignments are automatically graded and the student receives feedback immediately. The work also examines how it is possible to integrate these learning materials into everyday teaching, whether normal or distance learning.

Keywords

video course, Java, assignments, integration to the teaching, Moodle

Obsah

Úvod

Rád bych začal tím, že se podělím o to, co mě vlastně přivedlo k vypracování této práce. Když jsem v prvním ročníku střední školy začínal studovat programování, mnohokrát se mi stalo, že jsem látku, kterou jsem se naučil během hodiny, postupem času zapomněl, a když poté měl být test, musel jsem hledat způsoby, jak bych se danou látku doučil. Někdy si stačilo pouze přečíst textový materiál, který jsme na hodině ke každému novému tématu dostali. V případě, že jsem chyběl mi ovšem tyto materiály nestačily a musel jsem najít jiné způsoby, jak se látku doučit. Hledání tedy pokračovalo a začal jsem číst různé výukové materiály na internetu. Bohužel jsem se z nich moc nenaučil, jelikož jsem v té době nerad četl, a nedokázal jsem tedy u nich udržet tolik pozornosti, abych si z nich odnesl vše potřebné. Nakonec jsem skončil sledováním různých videí. Má úroveň angličtiny v té době byla celkem nízko, takže jsem sledoval pouze videa v mateřském jazyce. Problém byl, že zadarmo dostupná videa v češtině často nepokrývala vše, co jsme se na hodinách učili nebo téma pokrývala nedostatečně. Tehdy jsem si v hlavě začal pohrávat s představou, že by bylo skvělé, kdyby škola měla vlastní výuková videa, ze kterých bych se mohl naučit na jednom místě vše potřebné. Když jsme se na konci prvního ročníku bavili s mým učitelem programování, ing. Kahounem, a on sám řekl, že by bylo dobré, kdyby existovala školní výuková videa na programování, rozhodl jsem se definitivně, že tuto myšlenku realizuji.

Postupem času, když jsem prošel prvákem a druhákem, jsem si však uvědomil, že pouze videa nestačí. Programování je totiž praktický předmět, a pokud se ho někdo chce opravdu naučit, musí sám psát kód a sám přemýšlet nad různými problémy. Ke každému videu jsem tedy přidal také teoretické a praktické úlohy, na kterých si studenti procvičí získané znalosti okamžitě po zhlédnutí nějakého videa.

Původním cílem práce bylo tedy vytvořit ucelený systém, který by značně usnadnil samostudium žákům, kteří se rádi učí z videí a chtějí doučit nějaké téma z hodiny, ukotvit jejich znalosti pomocí cvičných úloh, a nebo je dokonce rozšířit. Jenže v průběhu vytváření práce se objevil COVID-19 a začala distanční výuka. Zde se objevil nový rozměr využití mé práce nejen jako prostředek pro samostudium, ale také jako prostředek pro alternativní formu distanční výuky. Práce tedy kromě vytvoření samotných materiálů poskytuje také návrhy, jak začlenit materiály do výuky jak prezenční, tak distanční a jak motivovat studenty, aby je využívali.

1 PRŮZKUM ZÁJMU STUDENTŮ O VÝUKU

Tato část práce má za cíl zjistit, zda je mezi studenty zájem o výuková videa a procvičovací úlohy a identifikovat, z jakých zdrojů studenti získávají nejvíce informací

Za tímto účelem byl všem čtyřem programátorským třídám, o celkovém počtu 120 studentů, poslán dotazník. Celkově na něj však odpovědělo pouze 33 z nich. Data získaná z průzkumu tedy pravděpodobně nemusí být úplně přesná, každopádně dávají nám jistý orientační rámec. Výsledky dotazníku budou použity jako motivace pro vytvoření materiálů a jako nástroj na návrh možných řešení začlenění těchto materiálů do výuky (viz kapitola 4).

1.1 Struktura dotazníku

V této části je zkoumána a zdůvodňována struktura dotazníku, který byl poslán studentům.

1.1.1 Identifikační otázky

Dotazník se skládal ze dvou identifikačních otázek:

- Do jakého ročníku docházíte?
  - 1
  - 2
  - 3
  - 4
- Programování se chci dále věnovat a mým cílem není pouze získat 4, abych prošel.
  - Ano
  - Ne, ale rád bych pomohl ke zlepšení kvality výuky programování na škole a mé odpovědi mohou být relevantní
  - Ne a mé odpovědi na otázky ohledně vylepšování výuky programování tedy nejsou relevantní

Studenti, kteří se totiž programování věnovat dále chtějí, odpovídají obecně na některé dotazy jinak než ti, kteří nechtějí, a díky této otázce je možné rozlišit odpovědi na další otázky podle tohoto kritéria. Dotazník byl totiž povolen k vyplnění i žákům, které sice programování nezajímá, ale mohou mít nápady na zlepšení výuky tak, aby je např. výuka začala bavit. V případě, že někdo zaškrtl poslední možnost, byl mu dotazník okamžitě ukončen.

Zároveň dotazník bere odpovědi jednotlivých ročníků jako rovnocenné. Ačkoliv se tato práce zaměřuje na tvorbu učebních materiálů pro první ročník, stejné materiály by mohly být použity i pro výuku ve vyšších ročnících.

1.1.2 Zkoumání jednotlivých částí výuky

Výuka programování na Gymnáziu Arabská je realizována v podobě prezenční výuky, popř. nyní ve formě distanční výuky ve výjimečné situaci. Je možné rozdělit výuku na čtyři oddělené části, které motivují žáka získávat znalosti. Jsou jimi:

- prezenční výuka = nepřerušovaný výklad o délce dvou vyučovacích hodin. Během ní se vysvětlují koncepty dle učebního plánu (Gymnázium, Praha 6, Arabská 14, 2016), popř. se procvičují úlohy. Její součástí jsou také testy, na jejichž zvládnutí je potřeba dávat na této části výuky pozor.
- distanční výuka = prezenční výuka na dálku přes internet, která je momentálně uplatňována kvůli situaci s nemocí COVID-19
- domácí úkoly = povinné úlohy na doma, na jejichž vyřešení je potřeba napsat zdrojový kód řešící daný problém.
- ročníkový projekt = aplikace, kterou musí student každý rok vytvořit. Více o něm (Kahoun, 2017)

Cílem této sekce je zjistit, která z těchto čtyř částí výuky donutí studenta naučit se nejvíce informací. Z výsledků poté bude jasné, které části žákům dávají nejvíce a je třeba jejich podíl ve výuce posílit a také ty, které je potřeba nějakým způsobem zlepšit. Kladené otázky byly:

- Dokážete udržet pozornost po celou prezenční dvouhodinovky?
  - Ano
  - Ne
- Dokážete udržet pozornost po celou dobu distanční dvouhodinovky?
  - Ano
  - Ne
- U každé z následujících částí výuky označte, kolik se toho z ní naučíte (prezenční výuka, distanční výuka, domácí úkoly, ročníkový projekt)
  - 1 – téměř nic
  - 2
  - 3
  - 4
  - 5
  - 6 – opravdu hodně

Z vlastních zkušeností vyvozuji hypotézu, že nejvíce se žák, který se o programování zajímá, naučí z ročníkového projektu, poté z domácích úkolů, následně z prezenční výuky a nejméně z distanční výuky.

1.1.3 Zkoumání zájmu o videa a dobrovolné procvičovací úlohy

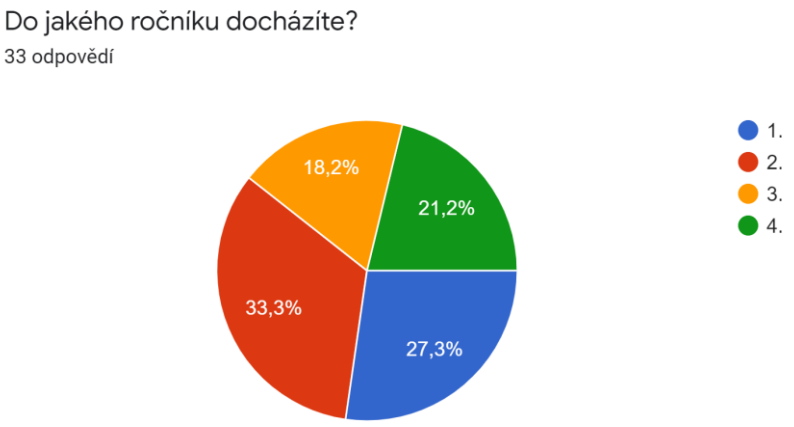
Tato část výzkumu je stěžejní, jelikož na jejích výsledcích závisí fakt, zda má vůbec smysl vytvářet tuto práci. Cílem je tedy zjistit, jak velká část studentů má zájem o výuková videa, dobrovolné domácí úkoly a také o některé návrhy, jak lépe strukturovat výuku. Dotazy tedy byly:

- Preferujete výuková videa nad textovými materiály?
    - Ano
    - Ne
- Koukáte někdy na výuková videa, protože jste nepochopili látku vysvětlovanou na hodině?
    - Ano
    - Ne
- Ocenili byste, kdyby vznikla videa pokrývající všechna témata, která učitel vysvětluje na hodině, ke kterým byste se mohli vrátit, když něčemu nerozumíte?
    - Ano
    - Ne
- Chtěli byste, aby se na začátku hodiny pustilo video pokrývající celé téma dané hodiny (5 - 20 minut) a ve zbylém čase se poté o videu diskutovalo a řešily by se cvičné úlohy k danému tématu? (V současnosti učitel výklad musí vyložit sám, což může být často velice zdlouhavé)
    - Ano
    - Ne
- Dávali byste na hodinách větší pozor, kdybyste mohli za úspěšné řešení úloh, které je na hodině procvičovány, získat kladné body?
    - Ano
    - Ne
- Ocenili byste možnost, kdyby ke každému většímu tématu byly dobrovolné domácí úkoly, za které by bylo možnost získat kladné body? (Jednalo by se zaprvé o praktické úlohy, kdy je vaším úkolem napsat kód a zadruhé o cvičné testy. U obou těchto příkladů by se vaše řešení automaticky otestovalo, oznámkovalo a byla by poskytnuta zpětná vazba)
    - Ano, více procvičování vítám jako prostředek na ukotvení znalostí
    - Ano, ale vypracoval bych je pouze za účelem vylepšení známky, jiný by nebyl.
    - Ne

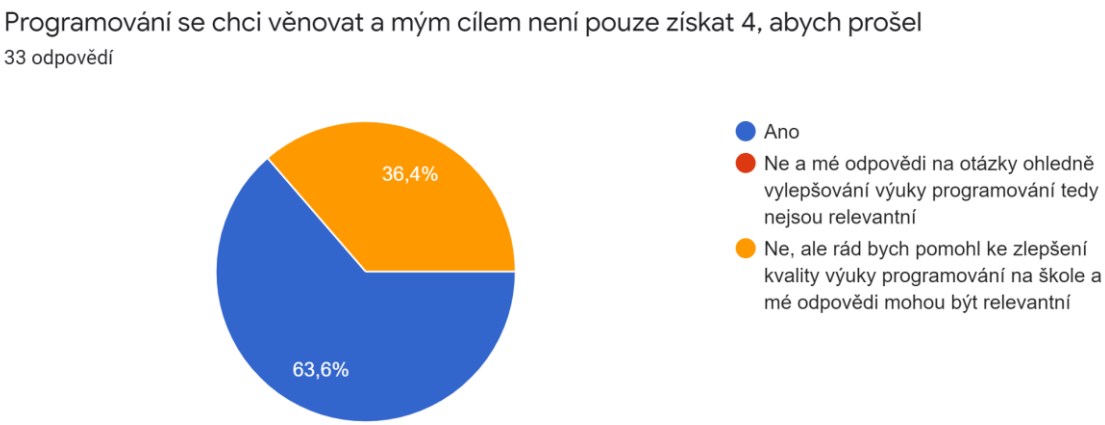
1.2 Analýza výsledků průzkumu

V této části jsou analyzovány odpovědi od 33 studentů na otázky v předchozích sekcích. Data jednotlivých odpovědí jsou analyzována a jsou z nich vyvedena nějaká tvrzení. To, jak výsledky tohoto průzkumu převést do praxe ovšem není předmětem této sekce, nýbrž kapitoly č. 4.

1.2.1 Identifikační otázky



Distribuce ročníků respondentů je vcelku vyvážená. Žáci prvního, druhého a třetího ročníku již mají zkušenost s některými výukovými videi z této práce, a tak jejich odpovědi mají svoji váhu.



Dominantní podíl respondentů tvoří žáci, kteří mají zájem o programování. Jelikož je počet respondentů vůči celkovému počtu studentů programování čtvrtinový, nemůžeme určit počet studentů na škole, které programování skutečně zajímá. Výsledky ovšem použijeme při vyhodnocení dalších otázek.

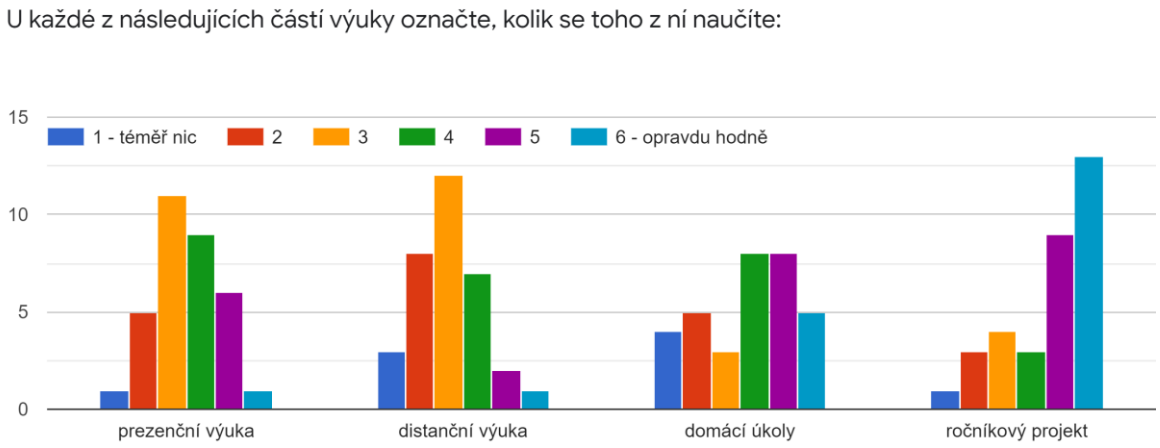
V případě, že není blíže specifikováno, studenti se zájmem o programování odpovídali stejným či hodně podobným způsobem, jako studenti, kteří o něj zájem nemají. V opačném případě budete výslovně upozorněni.

1.2.2 Zkoumání jednotlivých částí výuky

Když se na tyto dva grafy podíváme, vidíme značný rozdíl mezi prezenční a distanční výukou. V této části se vyskytoval značný rozdíl mezi odpověďmi studentů se zájmem o programování a bez něj. Zatímco při prezenční výuce udrží pozornost 12/21 studentů se zájmem, což je o něco více než polovina, tak pouze 2/12 studentů bez zájmu při ní dokáže udržet pozornost po celou dobu.

U distanční výuky dokáže udržet pozornost po celou dobu pouze 6/21 studentů se zájmem a 3/12 studentů bez zájmu. Závěrem porovnání je, že distanční výuka je pro studenty se zájmem mnohem horší než prezenční a pro studenty bez zájmu se téměř neliší.

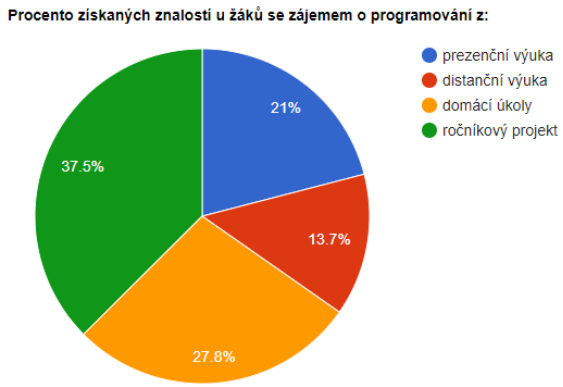
Návrhy na zvýšení procenta lidí, kteří udrží pozornost, najdete v kapitole 4.



Tato otázka je pravděpodobně nejdůležitější z celé druhé sekce. V kapitole o struktuře této sekce byla vyslovena hypotéza, že studenti se zájmem o programování budou největší část znalostí získávat z práce na ročníkovém projektu, poté domácích úkolech, pak prezenční výuce a poté a distanční výuce. Tato hypotéza byla vyslovena proto, že se domnívám, že programování je praktický předmět, kdy je pro ukotvení znalostí třeba, aby studenti sami bojovali s různými problémy a vymýšleli jejich řešení. Toto nejlépe poskytuje ročníkový projekt, který simuluje práci na reálné aplikaci. Prezenční výuka tedy studenta může uvést do problematiky, ovšem k ukotvení znalostí je třeba mnoho procvičování. Distanční výuku poté řadím za prezenční výuku hlavně kvůli všeobecnému předpokladu, že distanční výuka je méně efektivní než prezenční.

Abychom zjistili, kolik se toho student z jednotlivých částí naučí, vyhodnotíme tuto otázku pomocí váženého průměru. Vynásobíme počet odpovědí v daném stupni dané kategorie číslem kategorie a všechny získané výsledky dané kategorie sečteme. Pokud například u ročníkového projektu odpovědělo 13 lidí stupněm 6, 9 lidí stupněm 5, 3 lidé stupněm 4, 4 lidé stupněm 3, 3 lidé stupněm 2, 1 člověk stupněm 1, tak vážený průměr této kategorie bude 13*6+9*5+3*4+4*3+3*2+1*1=154. Čím vyšší toto číslo je, tím větší roli hraje při získávaní znalostí hraje.

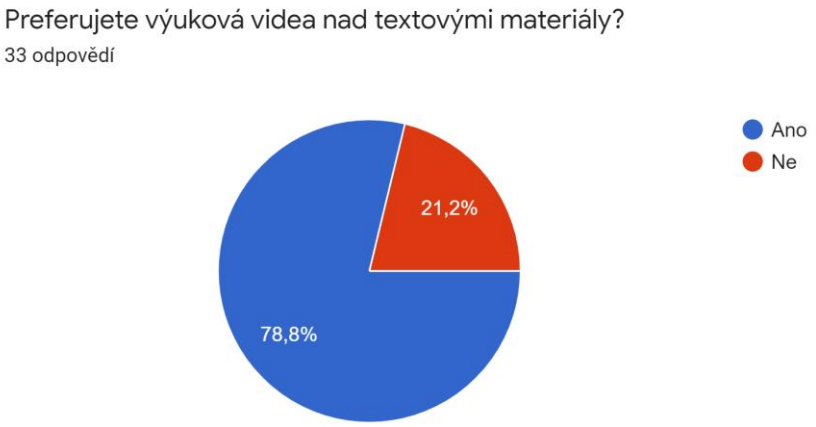
Rozdělíme si opět odpovědi podle zájmu studentů o předmět. Následující graf ukazuje, jaké procento zastávají jednotlivé části výuky při získávání znalostí. Tento graf získáme tak, že sečteme všechny vážené průměry a získáme procento každé kategorie z celkového součtu.



Vidíme, že v případě studentů se zájmem o předmět dominují 65% podílem ročníkový projekt a domácí úkoly. To jsou právě ty praktické části výuky, které byly zmíněny. Naše hypotéza se tedy pomocí všech tří otázek potvrdila. Pořadí, v jakém byly podíly jednotlivých částí na získávání znalostí předvídány, odpovídá skutečným datům.

Zároveň je zajímavé, že studenti bez zájmu o programování tento trend nesdílí. U nich jsou všechny části víceméně vyrovnané. Je to tím, že nemají onen zápal, který by je nutil řešit praktické problémy sami. Když je potřeba nějaké vyřešit, dost často někoho prostě jen požádají o pomoc, ne-li přímo o udělání úkolu.

1.2.3 Zkoumání zájmu o videa a dobrovolné procvičovací úlohy



Ze tří výše uvedených grafů je patrné, že důvod pro vypracování této práce určitě je. Tři čtvrtiny dotázaných preferuje výuková videa nad textovými materiály jako zdroj získávání

informací. Tím, proč tomu tak je, se budeme zabývat ve druhé kapitole. Drtivá většina z nich by poté byla ráda, kdyby vznikla videa pokrývající veškerou látku, kterou učitel vykládá na hodině, aby se k ní poté mohli samostatně vracet.

Tyto dvě části průzkumu slouží jako zpětná vazba na mé návrhy na zvýšení procenta lidí, kteří dokáží na hodině udržet pozornost. Více si k nim povíme v kapitole 4.

Ocenili byste možnost, kdyby ke každému většímu tématu byly dobrovolné domácí úkoly, za které by bylo možnost získat kladné body? (Jednalo by ..., oznámkovalo a byla by poskytnuta zpětná vazba)
33 odpovědí

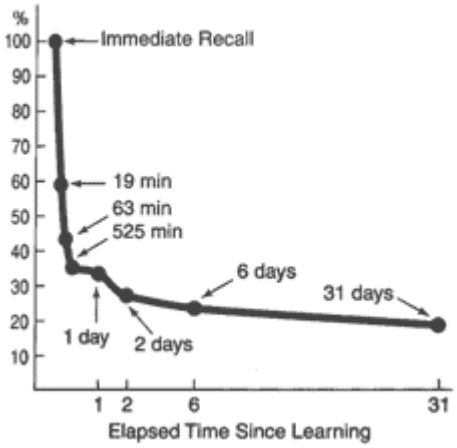
Z této poslední otázky je opět zřejmé, že žáci mají nejen zájem o výuková videa, ale i o procvičovací úlohy k nim. Žáci si tedy jsou vědomi, že nejvíce znalostí získávají praktickým procvičováním.

2 VIDEOKURZ

Tato část práce se věnuje samotnému videokurzu. Popisuje témata, která pokrývá, jeho strukturu, podrobné charakteristiky jednotlivých videí a zkoumá metodiku vytváření videí. Zkoumá také možné výhody výukových videí, díky nimž tři čtvrtiny dotázaných preferují při samostudiu videa před textovými materiály. Čtvrtá kapitola poté na tuto kapitolu navazuje možnými návrhy, jak videokurz začlenit do výuky.

2.1 Motivace

Většina dotázaných studentů nedokáže udržet pozornost po celou dobu prezenční dvouhodinové výuky a u distanční je tato část studentů ještě vyšší. I kdyby pozornost udržet dokázali, tak dle známé Ebbinghausovy křivky zapomínání (Ebbinghaus, 1885/1913) brzy zapomeneme drtivou většinu toho, co jsme se předtím naučili. Žáci tedy musí nějak své



Obrázek 1: Ebbinghausova křivka zapomínání

znalosti obnovovat a to hlavně pomocí praktických cvičení, např. domácích úkolů, jak jsme si ukázali. Aby ovšem byli studenti schopni úkol vypracovat, potřebují někde získat základní informace, jak k danému problému přistoupit. Tím mohou být právě tato videa.

2.1.1 Porovnání výukových videí s klasickou výukou

Jedna z výhod výukových videí oproti klasické výuce je, že student si může úsek, kterému nerozumí, pustit tolikrát, kolikrát potřebuje. To na prezenční klasické výuce možné není, jelikož kdyby se někdo stále ptal, zdržoval by hodinu. Když student něčemu naopak rozumí, může si daný úsek přehrát zrychleně či dokonce přeskočit, čímž krátí čas potřebný na zhlédnutí videa a tudíž déle udrží pozornost.

Zároveň platí, že když je video dobře udělané a třeba je předem napsaný scénář, tak na sebe jednotlivé části videa perfektně navazují. To nemusí nastat v klasické výuce, kdy učitel může často odběhnout od tématu či se nedostatečně jasně vyjadřovat. Tím naruší chápání studenta, který musí vynaložit větší snahu, aby danou látku pochopil. Díky odstranění těchto nedostatků je poté možné značně zkrátit délku výuky ze dvou hodin například pouze na intenzivních

dvacet minut. Jelikož student udrží pozornost tím snáze, čím je video kratší, dokáže u dobře udělaného videa tedy udržet pozornost lépe než u dvouhodinovky.

Výhodou také je, že studenti, kteří na hodině vůbec nebyli, si mohou jednoduše zjistit, co se na hodině dělalo. U každé hodiny je totiž napsáno její téma a student si toto téma vyhledá ve výukových videích a podívá se na něj.

2.1.2 Porovnání výukových videí s textovými materiály

Kromě prezenční výuky můžeme výuková videa porovnat také s textovými materiály. Tři čtvrtiny dotázaných uvedli, že preferují výuková videa nad textovými materiály. To pramení zaprvé ze způsobu, jakým náš mozek zpracovává informace. Je známý fakt, že někdo se efektivněji učí z videí a někdo zase z jiných zdrojů. Mozek některých lidí si totiž dokáže poradit s obrazovými asociacemi lépe než s abstraktními pojmy, které dostane z podoby textového materiálu.

Dalším důvodem ovšem může být také to, že čtení vyžaduje aktivní účast čtenáře, zatímco na video se člověk dívá pasivně, stačí mu jen poslouchat. Video tedy může být pro studenta mnohem pohodlnější, jelikož je pro něj snazší koukat na videa než číst materiály. Toto se může zdát jako nevýhoda, ale domnívám se, že opak je pravdou. Jak již bylo zmíněno a potvrzeno dotazníkem, programování je praktický předmět, nedá se naučit z žádných materiálů. Je třeba, aby student sám řešil praktické úlohy a prohluboval tak své znalosti. Není tedy potřeba, aby se student z materiálů učil programovat tak, že si něco zapamatuje, je potřeba, aby z nich mohl získat informace na vyřešení nějakého problému, a to je z videí alespoň dle mého názoru možné lépe než z textových materiálů.

Výuková videa jsou také mnohem lepší pro začátečníky z toho důvodu, že studentovi je přesně krok po kroku ukázáno, co má udělat a stačí mu tedy přesně opakovat kroky instruktora. Je mu přesně ukázáno, kdy má otevřít vývojové prostředí, kdy vytvořit novou třídu, kdy má kód spustit. I když prezentující nějakou akci nekomentuje, např. protože ji považuje za samozřejmost, tak je vždy v pozadí videa vždy vidět, co dělá. U textových materiálů toto tak být nemusí.

Jako nevýhody videí je možné uvést například jejich nemožnost učit se z nich na rušných místech či v místech, kde je omezený datový limit, jelikož na čtení kódu je potřeba mít alespoň rozlišení 480p, které spotřebovává dost dat. Pro tyto účely jsou lepší textové materiály. Pokud se tedy někdo z videí chce učit, musí si najít adekvátní podmínky.

2.2 Výroba videí

Prvním krokem k výrobě videokurzu bylo určit si tematický rozsah videokurzu. Tím se stal učební plán prvního ročníku předmětu programování na Gymnáziu Arabská. Byl rozšířen o některé zajímavé informace navíc a také o část látky z druhého ročníku, která pojednávala o tom samém tématu, jako látka v prvním ročníku a pouze jej rozšiřovala, např. téma Výjimky. Veškeré zdroje, které byly na tvorbu videí použity, jsou dostupné na konci práce

v bibliografii. U jednotlivých videí jsou poté v popisku videí uvedeny odkazy na zdroje daného videa. V případě, že by k danému videu nebyl zdroj, pak byly všechny informace v něm použité získány z výuky na škole či při samostudiu, a jedná se tak o všeobecně známou informaci.

Při jejich vytváření byl kladen velký důraz na to, aby se využívalo při vysvětlování pouze toho, co již bylo dříve vysvětleno, či co se nachází ve studijních předpokladech. Je třeba tedy na videa koukat popořadě, jinak student nemusí některým pojmům rozumět.

Zároveň byla učiněna snaha, aby videa nebyla příliš dlouhá, aniž příliš krátká. Ideální dobu jsem určil na 15 minut. Tohoto času dosáhla většina videí. Čtyři z témat byla ovšem natolik obsáhlá, že tento limit někdy odpovídající videa značně překročila. Při sledování takových videí je doporučeno v půlce učinit přestávku. Pokud by mělo být video delší než 30 minut, bylo rozděleno do více videí, např. téma Proměnné, Kolekce, Práce se soubory.

První fáze výroby videí probíhala od října 2019 do ledna 2020. Postup byl takový, že se nejprve uskutečnila jedna z dále popsaných částí pro všechna videa a poté se přešlo k dělání další části pro všechna videa. Nejprve tedy byly napsány všechny scénáře, poté udělán všechen dabing a poté přidání vizuálů a střih.

2.2.1 Scénář

Aby bylo jisté, že video pokryje všechny části tématu a zároveň se zabránilo zmatečným přeskakováním mezi tématy a přeřekům, bylo nutné ke každému videu napsat scénář. Ke psaní scénářů byl využit program Microsoft Word. Průměrný čas na napsání kvalitního scénáře včetně jeho upravování je kolem 3 hodin. Veškeré scénáře se nacházejí v příloze č. 3. Student scénář může využít jako textový materiál v případě, že z nějakého důvodu potřebuje nalézt jen několik málo konkrétních informací či z nějakého důvodu nemá možnost sledovat videa (slabý internet, rušivé prostředí). Jako samostatný studijní materiál je ovšem nedoporučuji, jelikož jsou přímo svázané s vizuální složkou a nejsou strukturovány do podoby tesxtového materiálu.

2.2.2 Dabing

Tato část zahrnovala předčítání scénářů a následné vylepšování kvality zvuku. K tomuto byl využit program Audacity a mikrofon Behringer C-1U. První částí bylo přečtení scénáře. V případě přeřeku nebo neobjevené chyby ve scénáři se chybná část předčítá znovu. Jakmile toto bylo dokončeno, vymazal se z audiostopy šum pomocí funkce zmenšení šumu. Poté se vylepšovala kvalita zvuku pomocí funkce kompresor s nastavením prahová hodnota -12 db, nežádoucí hluk -40 db, poměr 2:1, čas náběhu 0,2 sekund a čas uvolnění 1 sekunda. Výsledný zvuk se poté předá funkci Filter Curve, dříve nesoucí jméno ekvalizace. Té se nastaví tovární přednastavení na Bass Boost pro lepší kvalitu zvuku. Celý proces trvá průměrně 40 minut.

2.2.3 Tvorba vizuálů

Ke každému zvukovému záznamu bylo nutné vytvořit něco, na co se student bude při poslouchání dívat. V případě, že je ve videu zrovna něco programováno, tak je vizuálem záznam obrazovky s editorem Netbeans IDE. V případě, že je výklad teoretický, tak je vizuálem powerpointová prezentace, kdy se při mluvení přepínají slajdy. Tato část zabrala průměrně 2 hodiny.

2.2.4 Střih

Zvukovou stopu a vizuály je nutné sladit pomocí střihačského programu. Tím byl v tomto případě Davinci Resolve 16. Důraz byl kladen zejména na to, aby byly sladěny zvuková a vizuální stopa. To, o čem se mluví by tedy mělo být vždy nějak zobrazeno na obrazovce.

Po dokončení následuje takzvaný rendering. Ten převádí soubor vzniklý editací do spustitelné formy. Pro Youtube je tento formát přednastavený v nabídce programu. Videa jsou vytvořena pro FULL HD rozlišení.

Celý tento proces poté pro jedno video zabral průměrně 1 – 2 hodiny.

2.2.5 Zveřejnění a sběr zpětné vazby

Jakmile bylo video sestříháno, bylo okamžitě nahráno na YouTube kanál Programování na Arabské, odkaz v příloze, kde se na něj mohl kdokoliv podívat a psát zpětnou vazbu. Během této doby začínala distanční výuka a videa tedy byla zaslána jako experimentální nástroj pro výuku aktuálnímu prvnímu ročníku jako dobrovolné studijní materiály. Tato část probíhala až do listopadu 2020, kdy nastala další a poslední část výroby, tj. kontrola.

2.2.6 Kontrola

Předmětem této poslední části bylo projít celý videokurz znovu video po videu, najít v každém z nich všechny faktické chyby a opravit nedostatky, které byly nalezeny v páté části. Jakožto hlavní nedostatky videí byly určeny:

- vzácné faktické chyby
- občas příliš složitá souvětí, kterým lze těžko rozumět
- zvuková a vizuální stopa nesedí
- jednotlivé zvukové části na sebe u některých videí občas plynule nenavazují
- vizuály jsou málo interaktivní a je těžké se v nich vyznat
- kód zobrazovaný záznamem obrazovky nelze často přečíst na malých zařízeních

Na rozdíl od první fáze výroby videí bylo nyní k jejich opravě přistoupeno individuálně. Znamená to tedy, že vždy bylo opraveno jedno celé video od začátku do konce, nebyly najednou upravovány jednotlivé části. Díky tomu se podařilo předejít společným chybám ve videích, které by jinak nastaly.

Všem těmto problémům se tato část výroby věnovala. U každého videa byly upraveny powerpointové vizuály tak, že se věty, které tam dříve byly, nahradily klíčovými slovy, která se zobrazila pomocí animace rozplynutí, až když se o nich ve videu mluvilo. Byla také přiblížena obrazovka u všech záznamů obrazovky, aby byly lépe čitelné a podařilo se synchronizovat zvukové a vizuální stopy videí tak, že vizuál se na obrazovce objeví přesně v moment, kdy je zmíněn a nezačne se mluvit o dalším tématu, dokud tento vizuál nezmizí z obrazovky. U většiny videí tedy byly opakovány pouze kroky výroby 3 – 5, kroky 1 a 2 zůstaly z dřívějška. Časová náročnost se kvůli vyšší kvalitě vizuálů, animacím a novému střihu vyšplhala průměrně na 4 – 5 hodin času na video.

V případě, že se ve videu vyskytovaly faktické chyby, nepřesnosti či byla kvalita zvukového záznamu příliš špatná, došlo ke kompletnímu předělání videa. To znamená, že byly zopakovány kroky výroby 1 – 5. Toto se týká videí: 2, 3-1, 3-2, 3-3, 3-4, 5, 6, 10, 12-2, 14-3. U těchto videí byla také použita jiná technika na vylepšování zvuku. Vůbec se nepoužíval kompresor, ani ekvalizér, prostě se záznam jen zesílil a odstranil se šum. Videí tak zní mnohem přirozeněji.

Výsledné video je samozřejmě mnohem kvalitnější než předchozí verze, každopádně časová náročnost kompletního předělání se vyšplhala na 8 – 10 hodin za průměrné 15 minutové video.

2.3 Charakteristika jednotlivých videí

Nyní, když je výroba již dokončena, se na kanále Programování na Arabské nachází celkem 24 videí o celkové délce 6 hodin 10 minut a 58 sekund. Tato kapitola vám dá hrubou představu o tom, co se v jednotlivých videích vlastně nachází.

Videokurz je rozdělen do 14 základních tematických celků, kdy některé z nich byly natolik obsáhlé, že k nim bylo natočeno více videí.

2.3.1 Programování obecně

Video nejprve vysvětlí studentovi základní pojmy, které v programování bude potřebovat. Poté se přesune na vysvětlování toho, jakým způsobem je program spouštěn a vyhodnocován. Je tedy zmíněn rozdíl mezi zdrojovým a strojovým kódem a s tím související rozdíl mezi kompilovanými a interpretovanými jazyky. Dále žák zjistí, jaký je rozdíl mezi vyššími a nižšími jazyky a nakonec dostane nejzákladnější úvod do historie programování.

Toto video je jedno; bylo k učebnímu plánu přidáno navíc, jelikož věřím, že znalosti zde získané pomohou studentovi lépe pochopit látku v následujících videích.

2.3.2 O Javě obecně, první program

Ve videu je studentovi blíže představen jazyk Java. Seznámí se s jeho stručnou charakteristikou i historií. Následně je jasně definováno, co je to Java SE a jaké máme další platformy. Je také stručně představen JVM – virtuální stroj na spouštění bytokódu v Javě.

V druhé části videa už se pracuje s vývojovým prostředím a je napsán první program. Nakonec je blíže představeno samotné vývojové prostředí a čemu vlastně v souborovém systému jednotlivé složky a balíčky odpovídají.

2.3.3 Proměnné

Téma proměnných je velice obsáhlé a rozkládá se na celkem 6 videí:

1. **Úvod do proměnných**

Toto video je základním úvodem do problematiky proměnných. Zavádí pojmy deklarace, inicializace, primitivní a referenční datový typ a student si toto prakticky procvičí na příkladu. Dále je probrána viditelnost proměnných, tedy rozdělení na lokální a globální proměnné. Nakonec je zaveden pojem operace a student je seznámen s různými skupinami operátorů.

2. **Celočíselné proměnné**

První část videa je věnovaná vysvětlení způsobu, jak převádět z desítkové soustavy do dvojkové a zpět a je také odvozen efektivní způsob převodu za pomoci Hornerova schématu. Po porozumění této části se student dále naučí, co je to bit, byte a jak jsou vnitřně reprezentována kladná i záporná celá čísla pomocí jedničkového a dvojkového doplňku. Nakonec jsou prozkoumány způsoby, jak inicializovat proměnné v jiných soustavách než jen v desítkové.

3. **Reálné proměnné**

Cílem tohoto videa je, aby student rozuměl tomu, jakým způsobem jsou v Javě a obecně i v jiných programovacích jazycích reprezentována reálná čísla. Začátek videa se odehrává v IDE a je věnovaný různým problémům, které mohou při práci s nimi nastat. Po získání motivace získá student znalosti, které mu dovolí převádět reálná čísla z desítkové soustavy do dvojkové a zpět. Je opět odvozen efektivní způsob na základě Hornerova schématu. Toto je využito při ukázce reprezentace reálných proměnných pomocí standardu pro fixovanou desetinnou čárku. Následně je uveden standard pohyblivé desetinné čárky a je zdůvodněno, proč je lepší než standard pro fixovanou desetinnou čárku.

4. **Logické proměnné**

Na začátku videa je vysvětleno, co je to logická proměnná a je zaveden princip true, false. Následuje vysvětlení principů logických operací, tedy co to je negace, konjunkce a disjunkce. Dále jsou uvedeny nějaké technické zajímavosti, které souvisí a nakonec jsou do detailu probrány relační operace a nakonec bitwise operace.

5. **Vlastní metody**

Předmětem videa je zajistit, aby byl student schopen psát vlastní metody. Je vysvětlen způsob, jakým se metody volají, jak předávat parametry a jak vracet výsledky pomocí klíčového slova return. Nakonec jsou probrány modifikátory přístupu.

6. **Třída Math, BigInteger, BigDecimal**

Na začátku se student naučí používat třídu Math, která implementuje pokročilejší aritmetiku, jíž Java neobsahuje v podobě operátorů. Dále se naučí používat třídy BigInteger a BigDecimal, které umožňují reprezentovat čísla přesahující rozsah příslušných datových typů.

2.3.4 Podmínky a cykly

Na začátku videa je vysvětleno, co to je podmínka a jak ji v Javě pomocí klíčového slova if vytvořit. Student se dále naučí, jak vytvářet úplné podmínky a také else if podmínky. Bude schopný vhodně používat operátory AND a OR, aby spojoval více podmínek dohromady. Je mu také ukázáno, co je to switch a ternární výraz.

Tyto znalosti poté student využije v druhé části videa, kde jsou probrány cykly. Nejprve while a do while cyklus a poté for cyklus. Nakonec je také ukázáno, jak pracovat s vnořenými cykly.

2.3.5 Objektově orientované programování

Cílem videa je uvést studenta do základů objektově orientovaného programování. Na začátku ukázán motivační příklad, na kterém je jasně vidět důvod existence objektů. Následně je detailně rozebráno, jak objekt vytvořit, jakou roli hrají třídy, konstruktor a operátor new. Následuje vysvětlení, jaký je rozdíl mezi objekty a primitivními datovými typy co se paměťové reprezentace týče. Je také probráno klíčové slovo static a věci s ním spojené.

Konec videa se poté věnuje čtyřem základním pilířům OOP, kterými jsou zapouzdření, abstrakce, dědičnost a polymorfismus. Důkladně je však probráno pouze zapouzdření, jelikož všechny ostatní pilíře mají svá vlastní videa.

2.3.6 Dědičnost a polymorfismus

Video má za cíl vysvětlit třetí a čtvrtý pilíř OOP, tedy dědičnost a polymorfismus. Video okamžitě začíná motivačním příkladem, kdy vytváříme program, který by mohl reprezentovat předměty z Minecraftu. Nejedná se o způsob, jakým by se opravdu v Minecraftu předměty reprezentovaly, ale pouze o ilustrativní příklad na vysvětlení dědičnosti. Na tomto příkladu jsou vysvětleny všechny důležité pojmy jako rodič, potomek, překrytí metod, přetížení metod atd…

2.3.7 Pole

Toto video vysvětluje, jak pomocí pole reprezentovat N hodnot daného datového typu. Kromě samotného pole je zde zaveden nový typ cyklu – for each cyklus a je zde také rozebrána třída Arrays. Nakonec je ukázána práce s vícerozměrnými poli.

2.3.8 Datový typ char

Video ukazuje, jak jsou v počítači reprezentovány znaky. Jsou zavedeny pojmy znaková sada s příklady ASCII a UNICODE a kódování, např. UTF-16 či UTF-8. V části videa je poté rozebrán datový typ char, který slouží na reprezentaci znaků a jeho obalující třída Character.

2.3.9 Řetězce

Toto video využívá znalosti získané zhlédnutím předchozího videa. V první části je představena třída String, je naznačena její vnitřní implementace a je podrobně rozebráno, jak se s ní pracuje. Ke konci videa je poté zavedena třída StringBuilder a jsou uvedeny její využití.

2.3.10 Výčty

Ve videu je na příkladu planet slunečních soustav ukázána výhodnost využívání výčtů v našich programech. V průběhu videa jsou poté ukazovány všechny vlastnosti, kterými se oproti jiným třídám liší.

2.3.11 Abstrakce

Video navazuje na pátý a šestý díl, jelikož se zabývá druhým pilířem objektově orientovaného programování, tj. abstrakcí. Studenti zjistí, co to vlastně abstrakce je a jak jí můžeme v Javě dosáhnout. Tj. vytvořením abstraktní třídy, či rozhraní. Tyto dvě struktury jsou poté ve videu porovnávány. Na konec jsou detailně probrány třídy v Javě a s tím související zanořené datové typy.

2.3.12 Kolekce

Kolekce v Javě jsou poměrně rozsáhlé téma, a tak mu byly věnovány celkem 4 videa.

1. **Úvod do kolekcí, práce s ArrayListem**

Ve videu je představen Java Collections Framework a jaký je jeho účel. Následně video ukazuje, jak pracovat s ArrayListem, aby studenti viděli příklad nějaké kolekce. Nerozebírá ovšem jeho implementaci, to je až předmět dalšího dílu. Na konci videa je uvedena třída Collections a některé její důležité metody.

2. **Listy**

Video popisuje vnitřní implementaci ArrayListu, LinkedListu a dalších dvou struktur, jimiž jsou fronty a zásobník. Porovnává, které z nich vykonají které operace rychleji a na co se nejvíce hodí.

3. **Množiny**

Na začátku videa je zaveden nový typ kolekcí, množina. Je porovnána s Listy a je zdůvodněno, proč je dobré ji používat. Druhá část videa ukazuje, jak s ní pracovat a jaké jsou jednotlivé typy množin.

4. **Mapy**

Video nejprve definuje, co je to mapa a jak se liší od ostatních kolekcí. Následně ukáže, jak s mapami pracovat a ve druhé polovině zkoumá jejich vnitřní implementaci a složitost různých operací, které na mapách můžeme provádět.

2.3.13 Výjimky

Video vysvětlí nejprve, co to výjimka je a zdůvodní, proč výjimky potřebujeme. Následně ukáže, jak výjimky ošetřovat a nakonec ukáže, jak si můžeme vytvářet vlastní výjimky.

2.3.14 Práce se soubory

Stejně jako kolekce či proměnné, tak i práce se soubory je rozsáhlým tématem a jsou mu tedy věnována tři videa.

1. **Soubory**

Ve videu je ukázáno, jak se v programování reprezentují soubory, tj. ukazuje třídy File a Path. Neukazuje, jak z něj číst, či zapisovat. Dozvíte se z něj vše potřebné k tomu, abyste si mohli vytvořit vlastní jednoduchý průzkumník souborů. Na konci také ukazuje pomocnou třídu Files, na které je možné volat několik užitečných třídních metod.

2. **Čtení ze souboru**

Video ukazuje různé metody, jak získávat data ze souboru. Vysvětluje, co je to InputStream a jak pracovat s textovými soubory pomocí tříd BufferedReader či Scanner.

3. **Zápis do souboru**

Poslední video videokurzu hovoří o tom, jak různými způsoby vypisovat data do souboru. Charakterizuje je a určuje, které se nejvíce hodí na co. Mezi ukázané třídy určené na zápis do souborů patří FileOutputStream, FileWriter, BufferedWriter a PrintWriter. Navazuje také na předchozí video tím, že je jako příklad ve videu využit program na kopírování souborů.

3 SBÍRKA ÚLOH

Jak již bylo mnohokrát v práci řečeno, programování je praktický předmět, kdy se studenti učí hlavně řešením problémů. Zároveň bylo také v kapitole o výhodách a nevýhodách videokurzu řečeno, že pouhé sledování videí je pasivní záležitostí a nevyžaduje plnou soustředěnost. Z těchto dvou hlavních důvodů je nutné zajistit, aby měl student kromě výukových videí k dispozici zároveň sbírku úloh, na kterých si své znalosti procvičí. Nalézt ji můžete v přílohách v podobě odkazu na online Moodle instanci s kurzem Programování na Arabské.

3.1 Typy úloh

Sbírka úloh, která byla k videím vytvořena, obsahuje dva hlavní typy úloh. Jsou jimi testové a praktické úlohy. Jelikož původním cílem práce bylo vytvořit výukové materiály pro samostudium, nutnou podmínkou pro všechny úlohy ve sbírce je, že musí být automaticky testovatelné, aby student mohl ihned dostat zpětnou vazbu.

Původně měly být testové otázky vytvořeny v Google formulářích a praktické v nějakém jiném programu, jelikož standardem GA pro distanční výuku je používání G Suite. Použití Google formulářů by tedy po studentech nevyžadovalo učit se jiným technologiím. Google formuláře ovšem nepodporují některé požadované typy otázek, hlavně typ pro praktické úlohy, a tak byla sbírka úloh zpracována v LMS Moodle. Ten podporuje všechny níže zmíněné otázky a poskytuje výborné možnosti integrace do výuky, viz kapitola 4.

3.1.1 Testové úlohy

Účelem tohoto typu úloh je procvičit takové znalosti, které si student musí natvrdo zapamatovat nebo jdou jen těžko procvičit pomocí praktických úloh. Jedná se například o detaily vnitřních implementací funkcí Javy, historii programování, atd.

Pro každé video byl vytvořen právě jeden test obsahující otázky typů popsaných níže. Platí přitom, že test vždy obsahuje typ otázek č. 1 a okolo 6 otázek typu č. 2. Ostatní typy otázek se v testu mohou a nemusí nacházet. Typy testových otázek:

1. Doplňte slova do kontextu.

V textu byla vyjmuta některá slova a na jejich původní místa byly dosazeny selectboxy obsahující nabídku slov, které se na dané místo hodí. Pouze jedno z nich je ovšem to původní, tedy správné slovo.

2. Pravda, nebo lež.

Na řádku je napsán výrok, který je buď pravda, nebo lež. V závorce za výrokem se nachází buď slovo pravda, nebo lež podle toho, jaké hodnoty výrok nabývá.

3. Otázka s několika odpověďmi

U otázky se nachází N odpovědí, z nichž je právě jedna správná

4. Přiřaď definici k pojmu.

Na levé straně se nachází N pojmů očíslované 1 – N. Ke každému pojmu lze přiřadit právě jednu definici. Ve sbírce je zapsán každý pojem na stejném řádku jako jeho definice.

5. Stručná odpověď

Pod otázkou se nachází pole, kam student vloží odpověď na otázku. Toto je využito například při převodu čísel mezi soustavami.

3.1.2 Praktické úlohy

Úkolem studenta u tohoto typu úloh je napsat zdrojový kód, který řeší problém popsaný v zadání. Ke každému tematickému celku byly vytvořeny vždy alespoň 2 praktické úlohy, přičemž u některých rozsáhlejších celků, např. kolekce, byly vytvořeny 3. Při vytváření úloh byl kladen důraz hlavně na to, aby pro vyřešení dané úlohy nebylo nutné znát nějaké techniky, které se student ještě z videí nenaučil. Platí ovšem, že jakmile už byla látka studentovi vysvětlena, tak může být využita při řešení následujících úloh. Tím je docíleno toho, že si student své znalosti neustále opakuje.

Úlohy byly vytvořeny tak, aby nad nimi bylo nutné přemýšlet a hledat způsoby, jak využít znalosti získané z videí. Většinou tedy měly za úkol vytvořit něco, co má nějakou opravdovou funkci, např. variaci na šachy či program na reprezentaci tvarů v rovině, který implementuje základní pravidla geometrické operace. Snahou tedy bylo vyhnout se uměle vytvořeným problémům (např. klasický příklad Dog extends Animal pro výuku dědičnosti), které by měly za cíl pouze naučit studenta syntaxi jazyka či nějaké jeho funkce. Toto nemusí nutně platit pro úlohy k několika prvním videím, jelikož zde student ještě nezískal dost potřebných znalostí na to, aby byly vytvořeny “neuměle” úlohy.

Postupně ovšem student sbírá více a více znalostí a je tedy možné dávat obtížnější úlohy. Obtížnost je značena pomocí hvězdiček, kdy jedna hvězdička znamená, že úloha je velice jednoduchá a pět hvězdiček zase, že úloha je velice těžká. U každého názvu úlohy je poté toto hodnocení uvedeno, aby měl student jasno dříve, než se do ní pustí. Obtížnost byla určena na základě času, který mně samotnému trval na napsání vlastního řešení dané úlohy. Zatímco úlohy s jednou hvězdičkou trvají po zhlédnutí příslušných videí vyřešit zhruba 3 - 10 minut, pětihvězdičkové úlohy zaberou okolo 1,5 - 4 hodiny.

Každá úloha obsahuje následující části:

- vystihující název a stupeň obtížnosti;
- textové zadání, které jasně popisuje, co je cílem dané úlohy;

- rozhraní - kostra kódu, kterou student nemění a pouze do ní něco doplňuje. Jedná se například o deklarace tříd a metod, které jsou ze zadání nutné implementovat. Rozhraní také může obsahovat javadocy, které zadání upřesňují;
- řešení úlohy - každá úloha byla hned po jejím vymyšlení také vyřešena a jakmile student odešle svou odpověď, tak je mu ihned zobrazeno řešení ukázkové řešení.

Praktické úlohy jsou implementovány pomocí programu Coderunner (Lobb, 2014), což je plugin do Moodlu, s jehož pomocí je možné vytvářet v tomto LMS další typ otázek. Ten dovoluje učiteli vytvořit testovací kód, který bude spuštěn na vyhodnocení studentské odpovědi. Jeho hlavní použití je právě ve výuce programování. Při vytváření otázky typu Coderunner jsou specifikovány všechny výše zmíněné části úlohy a navíc také testovací kód, podle kterého se studentské řešení otestuje. Sbírka úloh je postavená tak, že tento kód volá na studentských odpovědích různé metody, předá jim předem definovaná data a zkontroluje, zda se vrácená hodnota rovná hodnotě, kterou testovací kód očekává. Může být buď plně napsán učitelem či může být využito zjednodušujících nástrojů, které Coderunner poskytuje. Je zde totiž možné specifikovat, co vlastně student má za úkol vypracovat. Coderunner poskytuje tři různé možnosti pro Javy a to jsou:

- Java method - odpovědí studenta je pouze jedna metoda
- Java class - odpovědí studenta je program obsahující jednu veřejnou třídu - jedná se o nejčastěji používaný typ ve sbírce
- Java program - studentskou odpovědí je kompletní program

Každý z těchto tří typů odpovědí dovoluje jinak zjednodušit jejich testování. Kromě vlastního testovacího kódu je totiž možné využít tzv. “test cases”, česky testové případy. Na každý takovýto případ je celá studentská odpověď znovu zkompilována a znovu spuštěna pro kód specifikovaný v případu. Díky tomu je možné studentskou odpověď testovat na více různých sadách dat, přičemž si můžeme být jisti, že předchozí případ nijak neovlivní další. Každá úloha obsahuje několik těchto test cases, kdy student pro každý může vidět, co má být jeho výsledkem. Pro tyto případy, které student vidí, se poté jeho kód vyhodnotí. Zároveň se tam také nachází skryté případy, které student nevidí, aby nemohl oklamat program, např. tím, že by detekoval, o jaký případ se jedná a na základě toho vypsal očekávaný výstup bez jakékoliv logiky programu.

Coderunner je možné nastavovat, např. lze stanovit maximální zdroje alokované programu pro jeho běh či je možné známkovat nejen na základě počtu splněných testů, ale i na základě rychlosti splnění požadavků.

Spouštění cizích zdrojových kódů představuje značné bezpečnostní riziko. Kvůli tomu nespouští Coderunner kód sám, ale využívá na to speciální sandbox server, v našem případě je to Jobe server (Lobb, 2014). Základní nastavení Coderunneru je posílat kódy na testování na Jobe server univerzity Canerbury. Počet testů za hodinu je ovšem limitován, a tak je nutné si spustit vlastní instanci Jobe serveru. Pro ještě vyšší bezpečnost je doporučené, aby tento server běžel na virtuálním počítači na serveru s firewallem. V nastavení Moodlu se poté nastaví IP adresa nového Jobe serveru.

4 ZAČLENĚNÍ DO VÝUKY

Jelikož sbírka úloh byla dokončena těsně před termínem letošní SOČky, byla v praxi zatím otestována pouze videa. První verzi videí dostali k dispozici studenti prvního ročníku 2019/20 a první ročník 2020/21 již má k dispozici druhou verzi videí. Zároveň bude v podobě dobrovolných domácích úkolů představena i sbírka úloh. Způsobům, jak správně začlenit výukové materiály z této práce, se věnuje tato kapitola.

4.1 Samostudium

Primární účel této práce bylo poskytnout studentům, kteří se lépe učí pomocí vizuálních materiálů, prostředek pro kvalitní samostudium. Vše je přitom dostupné pomocí Moodle kurzu. Jsou v něm k dispozici nejen testy a praktické úlohy, ale také odkazy na výuková videa roztříděná podle jednotlivých témat, čímž veškeré materiály centralizuje do jednoho místa.

Jednotlivým testům a praktickým úlohám lze nastavit způsob hodnocení. Když student bude postupně úlohy vypracovávat, bude jeho práce automaticky hodnocena a učitel mu za to poté může udělit kladné body, které se počítají přímo do předmětu. Aby se předešlo tomu, že jeden student vypracuje kód, rozešle ho celé třídě a díky tomu celá třída dostane kladné body, je možné zapojit do Moodlu plugin na kontrolu plagiátorství, např. JPlag (Tri LE, 2013). Díky tomu se zvýší motivace na vypracovávání úloh. Studenti vypracováním úloh nejen zdokonalí své znalosti, ale také mohou najít v úlohách chyby a upozornit na ně. Za každé takové nalezení chyby by student mohl být odměněn dalšími kladnými body.

4.2 Distanční výuka

Během vytváření této práce se objevil sekundární účel této práce, a tím je vylepšení distanční výuky. Jak bylo odhaleno v průzkumu, pouze 27% procent dotázaných dokáže po celou dobu distanční výuky udržet pozornost. Domnívám se, že je to hlavně z toho důvodu, že při distanční výuce značně klesá záživnost výkladu a zároveň proto, že učitel nemá kontrolu nad tím, co studenti dělají. Klidně student může odejít od počítače a učitel o tom vůbec neví, když ho zrovna nevyvolá.

Existuje několik tipů, které by mohly motivovat studenty, aby na distanční výuce dávali větší pozor. První z nich a asi nejdůležitější je, aby učitel se studenty komunikoval a to nikoliv se třídou, ale přímo s jednotlivci. Nejlépe je přímo náhodně vyvolávat. Tím je zajištěn základní předpoklad pro výuku, čímž je přítomnost studenta u obrazovky a nějaká základní pozornost.

Dalším tipem je na hodině nechat studenty vyřešit nějakou jednoduchou úlohu ze sbírky úloh (či nějaké jiné úlohy, kterou učitel vybere) a nabídnout za úspěšné řešení kladné body. Polovina dotázaných odpověděla, že by se na hodině více snažili, kdyby za řešení úloh na hodinách mohli získat kladné body. Rozhodně nemá cenu, aby učitel sám úlohy řešil, přednášel o postupu řešení a nechal studenty mezitím opisovat kód. Jediné, co tím procvičují, je psaní na klávesnici, jelikož jak studenti opisují, nedávají pořádně pozor na koncepty, které

jsou na úloze vysvětlovány nebo si je dlouhodobě nezafixují. Jediný případ, kdy bych toto doporučoval je při uvádění do tématu.

Posledním tipem je využití samotných výukových videí. Distanční výuka má nevýhodu, že je oproti prezenční mnohem složitější udržet pozornost. Pro učitele je navíc složitější komunikovat na dálku, jelikož před sebou nevidí studenty a nevidí tedy, zda jsou výkladem zaujati, či spíše znudění a už nedávají pozor. Pokud tedy učitel vykládá danou látku příliš pomalu, monotónně či jinak nudně, hrozí ztráta pozornosti. Je tu tedy možnost úplně z distanční výuky vyřadit výklad učitelem a místo toho dát v hodině prostor pro zhlédnutí videa. Tato doba by měla být minimálně 1,5x delší než samotné video, aby student mohl přehrát části, kterým nerozumí, vícekrát. Jak již bylo v sekci výhody videí probráno, video zajistí, že výklad bude konzistentní, nebude se odbíhat od tématu a nebudou v něm žádné přeřeky. Následně by byl vyčleněn čas, kdy by se stručně shrnul obsah videa (2-5 minut) a poté by následovalo cvičení, kde by studenti látku procvičovali. Pro tento tip se vyslovily v průzkumu dvě třetiny dotázaných a to ani nebylo upřesněno, zda se jedná o prezenční či distanční výuku. Je tedy možné předpokládat, že počet studentů, kteří by souhlasili, by byl ještě vyšší.

Jelikož jsou ovšem studenti velice odlišní, je nutné při zavádění každého z těchto tipů nutné se studenty zkonzultovat, zda o něco takového vůbec mají zájem. Může se např. sejít třída lidí, kde se většina lidí spíše učí z textových materiálů či jim více než videa vyhovuje klasický výklad.

4.3 Prezenční výuka

Pro prezenční výuku platí úplně to stejné jako pro distanční výuku. Sporné by zde bylo využití videí místo výkladu. V prezenční výuce totiž platí, že si učitel je schopen třídu hlídat a ví, jestli třídu výklad zajímá či nikoliv a dle toho může výklad upravit. Je tedy možné jednou za několik hodin pustit video jakožto zpestření výuky, každopádně prezenční výklad bych rozhodně nedoporučoval nahradit videy.

Co se ovšem týče procvičovacích úloh, tak si stojím za tím, co bylo napsáno v kapitole o distanční výuce.

ZÁVĚR

Práce naplnila všechny cíle práce, které jsem si stanovil. Byl natočen videokurz pokrývající veškerou látku prvního a části druhého ročníku předmětu programování na Gymnáziu Arabská. Ke každému tematickému okruhu byl poté vytvořen test a dvě praktické úlohy, které studentovi pomohou zapamatovat si informace získané z videí a využít je v běžné programátorské praxi. Na konci práce byly také představeny způsoby, jak by se daly efektivně tyto materiály začlenit do každodenní výuky, ať už probíhá ve formě prezenční, nebo distanční.

Z časových důvodů ještě nebylo možné otestovat sbírku úloh na studentech. Proces testování a sbírání zpětné vazby proto začne až v dubnu a na jeho základě bude sbírka úloh upravována. Tato upravovaná verze bude dostupná v druhém online Moodle kurzu Programování prvního ročníku (upravovaná verze), na který naleznete odkaz v příloze. Dalším způsobem, jak v práci pokračovat by bylo vytvořit další úlohy a videa pokrývající látku vyšších ročníků.

ZDROJE PRO VYTVOŘENÍ PRÁCE, VIDEÍ A SBÍRKY ÚLOH

https://www.biography.com/scholar/ada-lovelace

Bodnar, J. (19. 7 2013). *Difference between Java/SE/EE/ME.* Načteno z stackoverflow.com: https://stackoverflow.com/questions/2857376/difference-between-java-se-ee-me

Čápka, D. (7. 6 2012). *itnetwork.cz.* Načteno z úvod do objektově orientovaného programování v Javě: https://www.itnetwork.cz/java/oop

Čápka, D. (20. 5 2012). *Základní konstrukce jazyka Java*. Načteno z itnetwork: https://www.itnetwork.cz/java/zaklady

Gymnázium, Praha 6, Arabská. (1. 11 2016). *Informatika a informační a komunikační technologie.* Načteno z gyarab.cz: https://www.gyarab.cz/media/svp/ICT_Programovani_pg_zuNI5wA.pdf

Hanák, D. (2012). *Hornerovo schéma.* Načteno z itnetwork.cz: https://www.itnetwork.cz/navrh/algoritmy/algoritmy-matematicke/algoritmus-matematicke-hornerovo-schema

Herout, P. (14. 6 1999). *kiv.zcu.cz.* Načteno z Kódy UNICODE a UTF-8: http://www.kiv.zcu.cz/~herout/pruzkumy/unicode/unicode.html

Joseraj, J. (26. 8 2019). *dzone.com.* Načteno z The JVM Architecture Explained: https://dzone.com/articles/jvm-architecture-explained

Kahoun, D. (29. 1 2012). Programování, 1.E. Praha, Česká Republika.

kartik. (nedatováno). *Bitwise operators in Java.* Načteno z geeksforgeeks.org: https://www.geeksforgeeks.org/bitwise-operators-in-java/

Krubová, K. (2000). *Historie programovacích jazyků.* Načteno z fi.muni.cz: https://www.fi.muni.cz/usr/jkucera/pv109/2000/xkrubova.htm

Kříž, P. (2002). *Vývoj programování a programovacích jazyků*. Načteno z fi.minu.cz: https://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm

Mička, P. (nedatováno). *ASCII tabulka.* Načteno z algoritmy.net: https://www.algoritmy.net/article/89/ASCII-tabulka

Mička, P. (nedatováno). *Zanořené typy.* Načteno z algoritmy.net: https://www.algoritmy.net/article/24773/Zanorene-typy-15

Mittal, S. (2018). *8 Difference between BufferedReader and Scanner in Java.* Načteno z javahungry.blogspot.com: https://javahungry.blogspot.com/2018/12/difference-between-bufferedreader-and-scanner-in-java-examples.html

Mosh, P. w. (29. 3 2018). *Object-oriented Programming in 7 minutes.* Načteno z youtube.com: https://www.youtube.com/watch?v=pTB0EiLXUC8

Nitsdheerendra. (nedatováno). *Vector vs ArrayList in Java.* Načteno z geeksforgeeks.org: https://www.geeksforgeeks.org/vector-vs-arraylist-java/

Oracle. (nedatováno). *Java Platform Standard Edition 7 Documentation.* Načteno z docs.oracle.com: https://docs.oracle.com/javase/7/docs/index.html

paul, j. (2. 2 2017). *How HashMap works in Java.* Načteno z javarevisited.blogspot.com: https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html

Štechmüller, P. (24. 11 2018). *kolekce a proudy.* Načteno z itnetwork.cz: https://www.itnetwork.cz/java/kolekce-a-proudy

Ebbinghaus, H. ((1885/1913). *Memory: A Contribution to Experimental Psychology.* Načteno z studium-psychologie.cz.

Gymnázium, Praha 6, Arabská 14. (1. 11 2016). *Programování.* Načteno z gyarab.cz: https://www.gyarab.cz/media/svp/ICT_Programovani_pg_zuNI5wA.pdf

Kahoun, D. (2. 7 2017). *Ročníkový projekt.* Načteno z vyuka.gyarab.cz: https://vyuka.gyarab.cz/kahoun/download/rocnikovy_projekt_pozadavky.pdf

LE, T. (2013). *Plagiarism: Source-code Plagiarism Plugin.* Načteno z moodle.org: https://moodle.org/plugins/plagiarism_programming

Lobb, R. (2014). *Coderunner.* Načteno z github.com: https://github.com/trampgeek/moodle-qtype_coderunner

Lobb, R. (2014). *Jobe.* Načteno z github.com: https://github.com/trampgeek/jobe

SEZNAM PROGRAMŮ

- Microsoft Word, Powerpoint
- Audacity 2.3.3
- Davinci Resolve 16
- GIMP 2.10.14
- OBS Studio 24.0.3
- JDK 8u231
- Netbeans 8.2 IDE

SEZNAM OBRÁZKŮ

PŘÍLOHA 1: ODKAZ NA YOUTUBE KANÁL

PŘÍLOHA 2: ODKAZ NA FUNKČNÍ MOODLE KURZY

PŘÍLOHA 3: SCÉNÁŘE K VIDEÍM

PŘÍLOHA 4: POSUDEK VEDOUCÍHO PRÁCE

PŘÍLOHA 5: POSUDEK OPONENTA A ODPOVĚĎ NA OTÁZKY OPONENTA