Objektovo-orientované programovanie
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 28. jún 2017

(vyplňte tlačeným písmom)
Priezvisko:

Meno:

1 b	
2 b	
3 b	

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Skúška trvá 70 minút.
Odpovede na otázky 1–12 vpíšte do tabuľky. Pri týchto otázkach sa hodnotia len odpovede v tabuľke (bez postupu). Odpoveď musí byť jednoznačná a čitateľná, inak má hodnotu 0 bodov.
V otázkach s ponúknutými odpoveďami je len jedna možnosť správna – do tabuľky vpíšte len písmeno, ktorým je označená odpoveď, ktorú vyberáte.
Odpoveď na otázku 13 píšte výlučne na list, na ktorom sa nachádza jej znenie.
Poškodený list nebude uznaný.

1. (1 b) V jazyku C# zástupca (delegate) reprezentuje
(a) hocijakú funkciu
(b) ukazovateľ na objekt
(c) vlastnosť zodpovedajúceho typu
(d) hocijakú vlastnosť
(e) funkciu zodpovedajúcich typov parametrov a návratovej hodnoty

2. (1 b) Bodový prierez v jazyku AspectJ
(a) okrem iného, mení hodnoty parametrov metód
(b) rozdeľuje program v stanovených bodoch
(c) pridáva nové prvky do jestvujúcich tried
(d) zachytáva miesta, v ktorých program môže byť ovplyvnený
(e) ovplyvňuje program v stanovených miestach

3. (1 b) Ak by ste v jazyku C++ implementovali sčítavanie a odčítavanie farieb, na zjednodušenie zápisu týchto operácii by ste použili
(a) virtuálne funkcie
(b) preťaženie funkcií
(c) preťaženie operátorov
(d) virtuálne operátory
(e) statické funkcie

4. (1 b) Z hľadiska objektovo-orientovaného prístupu rozsiahle používanie statických metód
(a) nie je žiaduce, lebo nepodporujú zapuzdrenie
(b) nie je žiaduce, lebo sa nedajú prekonať
(c) je žiaduce, lebo sa vykonajú rýchlejšie
(d) je žiaduce, lebo umožňujú polymorfizmus
(e) nie je žiaduce, lebo sa nededia

5. (1 b) Je možné V Jave pristúpiť k triede definovanej v inom balíku bez importovania tohto balíka?
(a) áno
(b) áno, ale nedá sa od nej dediť
(c) nie
(d) áno, ale trieda musí byť uložená v tom istom adresári
(e) áno, ale trieda musí byť v tom istom súbore

6. (1 b) V jazyku C++ možno vytvoriť nový objekt
(a) nie z triedy, ale iba zo šablóny
(b) výlučne pomocou operátora **new** ako v Jave
(c) iba z tried, ktoré neobsahujú virtuálne funkcie
(d) pomocou operátora **new** ako v Jave alebo definovaním premennej typu danej triedy
(e) výlučne definovaním premennej typu danej triedy

7. (2 b) V danom softvérovom riešení je potrebné zastrešiť rôzne spôsoby riešenia toho istého problému. Ktorý návrhový vzor by ste použili?
(a) Strategy
(b) Composite
(c) MVC
(d) Visitor
(e) Observer

8. (2 b) Daný je nasledujúci program v Jave:

```
class A {
    private static int a = 'a', b = 'b';
    public static void f() {
        synchronized(A.class) {
            if (a == 'a') {
                a = 'b';
                b = 'a';
            } else {
                a = 'a';
                b = 'b';
            }
            if (a == b)
                System.out.println("=");
        }
    }
    public static void g() {
            if (a == b)
                System.out.println("=");
    }
    public static void main(String[] args) {
        A a = new A();
        (new B(a)).start();
        (new B(a)).start();
    }
}

class B extends Thread {
    A a;
    public B(A a) {
        this.a = a;
    }
    public void run() {
        for (int i = 1; i < 100000; i++)
            a.f();
            a.g();
    }
}
```

Uveďte kvalifikované názvy metód, pri ktorých je *nutné* uviesť modifikátor **synchronized**, aby bolo zaručené, že sa znak = nikdy nevypíše, ak také sú.

9. (2 b) Čo všetko sa vypíše prostredníctvom príkazov System.out.print() po spustení nasledujúceho programu v Jave (po jeho úspešné alebo neúspešné ukončenie)?

```
class E extends Exception {}

class M {
    public void f(int i) throws E {
        System.out.print(i);

        if (i < 0)
            throw new E();
    }
    public void g(int i) throws E {
        try {
            f(i);
        } catch (E e) {
            throw e;
        } finally {
            System.out.print("F");
        }
    }
    public static void main(String[] args) throws E {
        new M().g(1);
        new M().g(−1);
    }
}
```

10. (2 b) Hra v Jave je ovládaná stlačením zodpovedajúcich tlačidiel takto:

```
switch (o)
case 'a':
    if (player1.getEnergy() < 10)
        player2.addObject(player1.takeObject());
    else
        player1.addEnergy(player2.getEnergy());
    break;
case 'b':
...
```

Z hľadiska kvality objektovo-orientovaného návrhu by bolo vhodné

(a) použiť zapuzdrenie
(b) použiť polymorfizmus
(c) implementovať ovládanie myšou namiesto klávesnicou
(d) ponechať všetko tak, ako je
(e) vyčleniť kód jednotlivých prípadov (case) do zodpovedajúcich metód

11. (3 b) Čo sa vypíše po spustení nasledujúceho programu v Jave?

```
interface I {
    void m();
}
abstract class P implements I {
    public void m() {
        System.out.print("P");
    }
}
class Q extends P {
    public void m() {
        System.out.print("Q");
    }
}
class R extends Q {
    public void m() {
        super.m();
        System.out.print("R");
    }
}
class C {
    public static void exe(I... a) {
        for (I e : a)
            e.m();
    }
    public static void main(String[] args) {
        P x = new R();
        P y = new Q();
        R z = new R();
        I w = (I) new Q();

        exe(x, y, (P)z, (Q)z, w);
    }
}
```

12. (3 b) Trieda Kruh je odvodená od triedy Elipsa, pričom dedí atribúty, ktorými sú reprezentované (dve) ohniská elipsy a dĺžky jej (dvoch) polomerov. Metódy na nastavenie súradníc ohnísk a dĺžky polomerov sú prekonané tak, aby súradnice oboch ohnísk a dĺžky oboch polomerov pri zmene hociktorého z nich mali rovnakú hodnotu (čiže pri zmene jedného, zmení sa aj druhé). Týmto sa predpoklady a dôsledky pôvodnej metódy zoslabujú, zosilňujú alebo sa nemenia? Je týmto dodržaný Liskovej princíp substitúcie (LSP)?

Odpovedzte vo forme: *predpoklady / dôsledky / LSP*. Položky *predpoklady* a *dôsledky* nahraďte jednou z možností *zoslabujú sa*, *zosilňujú sa* alebo *nemenia sa*. Položku *LSP* nahraďte jednou z možností *dodržaný* alebo *nedodržaný*.

(vyplňte tlačeným písmom)

Priezvisko:

Meno:

13. (10 b) Virtuálny priestor pozostáva z poprepájaných buniek. Bunka môže byť nečleniteľná alebo členiteľná. Členiteľná bunka môže obsahovať ďalšie nečleniteľné a členiteľné bunky. Bunky môžu byť prepojené bez ohľadu na to, či sú nečleniteľné alebo členiteľné. Pri každej bunke je možné získať zoznam buniek, do ktorých možno z tejto bunky vstúpiť, čo sú bunky, s ktorými je daná bunka priamo prepojená, a – v prípade členiteľnej bunky – bunky, z ktorých daná bunka bezprostredne pozostáva (prvá úroveň).

Navrhnite a implementujte (t. j. napíšte príslušný program) v Jave zodpovedajúce objektovo-orientované riešenie zohľadňujúce princípy objektovo-orientovaného programovania. Využite pritom najvhodnejší z návrhových vzorov Strategy, Observer, Visitor a Composite.

Základný návrh predložte vo forme náčrtu diagramu tried v UML, ktorý bude obsahovať najvýznamnejšie vzťahy, operácie a atribúty. Zoberte pritom do úvahy návrhový vzor. Viditeľnosť nie je potrebné uvádzať.

V implementácii sa sústreďte sa na aplikačnú logiku – GUI nie je predmetom otázky. Taktiež, použité algoritmy nemusia byť optimálne.

Identifikujte explicitne prvky, ktorými sú modelované a implementované roly aplikovaného návrhového vzoru, a vysvetlite, prečo ste ho aplikovali. Poskytnite príklad použitia, v ktorom vytvoríte príslušné objekty a spustíte ich interakciu.

Objektovo-orientované programovanie
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 28. jún 2017
30

1 e

2 d

3 c

4 b

5 a

6 d

7 a

8 A.g()

9 1F-1F

10 e

11 QRQQRQRQ

12 nemenia sa / zoslabujú sa / nedodržaný

V poslednej otázke mal byť aplikovaný vzor Composite. Bunka ako taká bola v role Component (rozhranie alebo abstraktná trieda), členiteľná bunka v role Composite, a nečleniteľná bunka v role Leaf (triedy odvodené od triedy reprezentujúcej bunku). Operácia výpisu alebo vrátení spojení mala byť implementovaná tak, aby sa dala uplatniť transparentne aj nad členiteľnou, aj nad nečleniteľnou bunkou (mala byť prítomná už v bunke, a v členiteľnej a nečleniteľnej bunke prekonaná).

Aj členiteľná, aj nečleniteľná bunka mali obsahovať evidenciu (napr. zoznam alebo pole) buniek, s ktorými sú spojené. Členiteľná bunka mala navyše obsahovať evidenciu buniek, z ktorých pozostáva. Operácia výpisu alebo vrátení spojení pri nečleniteľnej bunke mala vrátiť len bunky, s ktorými je daná bunka spojená, kým pri členiteľnej bunke k tomu mali pribudnúť aj bunky, z ktorých táto pozostáva.

Otázka bude hodnotená podľa nasledujúceho kľúča:

- zabezpečenie základnej funkčnosti – 4 b
- návrh vzhľadom na princíp otvorenosti a uzavretosti kódu a adekvátne použitie zapuzdrenia – 6 b

Pri hodnotení obidvoch časti bude zohľadnené poskytnuté vysvetlenie (vrátane príkladu použitia) a zodpovedajúci diagram tried vo výške približne 10–20% hodnotenia príslušnej časti.