Objektovo-orientované programovanie 2018/19 **A**

doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU

Skúška – 7. jún 2019

(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++ sa preťažením operátora

(a) mení význam tohto operátora pre všetky typy operandov
(b) mení význam tohto operátora pre daný typ operandov
(c) význam tohto operátora vôbec nemení
(d) mení význam tohto operátora zásahom do kompilátora
(e) mení význam tohto operátora zásahom do linkera

2. (1 b) V jazyku C++ sa deštruktor

(a) musí uviesť v každej triede
(b) neuvádza nikdy
(c) musí uviesť v triedach, ktoré sú odvodené od viacerých iných tried
(d) musí uviesť v triedach, v ktorých sa pri tvorbe ich objektov dynamicky alokuje pamäť
(e) musí uviesť v triedach, ktoré obsahujú virtuálne metódy

3. (1 b) Jazyk C# zjednodušuje zabezpečenie zapuzdrenia prostredníctvom

(a) delegátov
(b) udalostí
(c) vlastností
(d) šablón
(e) makier

4. (1 b) Serializácia v Jave slúži na

(a) ochranu údajov
(b) uloženie zdrojového kódu
(c) agregáciu údajov
(d) synchronizáciu údajov
(e) uchovanie údajov

5. (1 b) V Jave to, že je metóda generická znamená, že

(a) nemá telo
(b) nemá parametre ani návratovú hodnotu
(c) má aspoň jeden parameter alebo návratovú hodnotu typu Object
(d) neimplementuje všetky detaily, ale časť jej kódu sa automaticky generuje na základe parametrov
(e) má typ aspoň jedného parametra alebo návratovej hodnoty zastúpený premennou

6. (1 b) V jazyku AspectJ je možné bez úprav jestvujúcich typov

(a) plne spravovať vykonanie metódy
(b) zabezpečiť vykonanie kódu pred, po alebo namiesto metódy, ale nie aj podmienečne vykonať metódu
(c) zabezpečiť vykonanie kódu pred alebo po metóde, ale nie aj namiesto metódy
(d) pridať nové metódy, ale nie aj ovplyvniť správanie jestvujúcich metód
(e) pridať nové atribúty, ale nie aj metódy

7. (2 b) Daný je nasledujúci program v Jave:

```
class S1 extends Thread {
    C c;
    public S1(C c) {
        this.c = c;
    }
    public synchronized void run() {
        synchronized(this) {
            for (int i = 0; i < 10000; i++)
                c.s1();
        }
    }
}

class S2 implements Runnable {
    C c;
    public S2(C c) {
        this.c = c;
    }
    public synchronized void run() {
        for (int i = 0; i < 10000; i++)
            c.s2();
    }
}

class C {
    private char o1 = 0, o2 = 0;

    public synchronized void s1() {
        if (o1 != o2)
            System.out.println("s1");
        o1 = 1;
        o2 = 1;
    }

    public synchronized void s2() {
        synchronized(this) {
            if (o1 != o2)
                System.out.println("s2");
            o1 = 2;
            o2 = 2;
        }
    }

    public static synchronized void main(String[] args) {
        C c = new C();
        new S1(c).start();
        new Thread(new S2(c)).start();
    }
}
```

Pri ktorých metódach je možné odstrániť modifikátor **synchronized**, aby stále bolo zaručené, že sa nič nikdy nevypíše (uveďte ich v zápise: `Trieda.metóda()`)?

8. (2 b) Potrebné je zabezpečiť, aby objekty, ktoré sa správajú podľa zmien stavu iného objektu, o týchto zmenách vedeli a mohli pribúdať bez potreby modifikovať tento objekt. Ktorý návrhový vzor by ste použili?

(a) Visitor
(b) Observer
(c) MVC
(d) Composite
(e) Strategy

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 C {
    public void f(int n) throws E {
        if (n < 1)
            throw new E();
    }

    public void g(int n) {
        System.out.print(n);

        try {
            f(n);
        } catch (E e) {
            System.out.print("E");
        } finally {
            System.out.print("F ");
        }
    }

    public static void main(String[] args) {
        new C().g(1);
        new C().g(0);
        new C().g(1);
        new C().g(0);
    }
}
```

10. (2 b) Hra v Jave obsahuje nasledujúci kód:

```
class Player {
    private int energy;
    private int lives;
    ...
    public void energyToLives() {
        acquiredLives = energy/100;
        lives += acquiredLives;
        energy -= energy/100 * 100;
        GameGUI.mainWindow.numberOfLives.setText(
            Integer.toString(lives));
    }
    ...
}
```

Hlavný problém v tomto kóde z hľadiska objektovo-orientovaného návrhu je to, že

(a) vnútorná logika sa mieša s používateľským rozhraním
(b) atribúty sú **private** a odvodeným triedam nebudú prístupné
(c) kód na prepočítanie energie na životy nie je súčasťou zodpovedajúceho prijímača
(d) metóda mení dva atribúty
(e) kód na prepočítanie energie na životy nie je súčasťou triedy okna hry

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

```
class A {
    public void x() {
        System.out.print("Ax");
    }
    public static void y() {
        System.out.print("Ay");
    }
}
class B extends A {
    public void x() {
        super.x();
        System.out.print("Bx");
    }
    public static void y() {
        A.y();
        System.out.print("By");
    }
}
class C extends B {
    public void x() {
        System.out.print("Cx");
    }
    public static void y() {
        System.out.print("Cy");
    }
}
class M {
    public static void main(String[] args) {
        A o1 = new B();
        C o2 = new C();
        A o3 = new A();
        B o4 = new B();
        B o5 = new C();

        ((B) o1).x();
        ((B) o1).y();
        System.out.print(" ");

        ((A) o2).x();
        ((A) o2).y();
        System.out.print(" ");

        o3.x();
        o3.y();
        System.out.print(" ");

        o4.x();
        o4.y();
        System.out.print(" ");

        ((A) o5).x();
        ((A) o5).y();
    }
}
```

12. (3 b) Trieda, ktorá reprezentuje špeciálny dokument, je odvodená od triedy, ktorá reprezentuje všeobecný dokument. Metóda na podpísanie dokumentu, ktorej jedným z parametrov je signatár, je v špeciálnom dokumente prekonaná a vyžaduje, aby signatár dokumentu bol zo zoznamu povolených signatárov. Týmto sa dôsledky a predpoklady pôvodnej metódy zoslabujú, zosilňujú alebo nemenia? Je týmto dodržaný Liskovej princíp substitúcie (LSP)?

Odpovedzte vo forme: *dôsledky / predpoklady / LSP*. Položky *dôsledky* a *predpoklady* 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) V zjednodušenej počítačovej reprezentácii mesto pozostáva zo stavebných objektov, ktoré predstavujú akékoľvek budovy alebo samostatné celky v rámci týchto budov. Každý takýto samostatný celok sa môže členiť na ďalšie samostatné celky a miestnosti, ktoré sa už ďalej nečlenia.

Používateľ môže vstúpiť do stavebného objektu, pričom systém vypíše zoznam identifikátorov stavebných objektov a miestností, z ktorých tento stavebný objekt pozostáva. V prípade, že používateľ vstúpil do miestnosti, systém vypíše iba identifikátor tejto miestnosti.

Navrhnite a implementujte 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 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.

Odpoveď bude hodnotená podľa nasledujúceho kľúča:

- zabezpečenie základnej funkčnosti – 4 b
- kvalita a flexibilita objektovo-orientovaného návrhu – 6 b

Objektovo-orientované programovanie 2018/19 **A**
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – 7. jún 2019
30

1 b

2 d

3 c

4 e

5 e

6 a

7 C.main(), C.s2(), S1.run(), S2.run()
(keďže pri metóde C.s1() omylom nebol uvedený modifikátor **synchronized**, akceptovaná je aj odpoveď typu „nedá sa dosiahnuť“)

8 b

9 1F 0EF 1F 0EF (akceptovaná je aj odpoveď bez medzier)

10 a

11 AxBxAyBy CxAy AxAy AxBxAyBy CxAy (akceptovaná je aj odpoveď bez medzier)

12 nemenia sa / zosilňujú sa / nedodržaný

V poslednej otázke mal byť aplikovaný vzor Composite. Stavebné objekty by boli v role Composite, a miestnosti v role Leaf. Obidva druhy objektov by boli zastrešené rolou Component.