Objektovo-orientované programovanie 2021/22
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 20. jún 2022

(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) Jazyk C# zjednodušuje zabezpečenie zapuzdrenia prostredníctvom
(a) delegátov
(b) udalostí
(c) makier
(d) vlastností
(e) šablón

2. (1 b) Serializácia v Jave slúži na
(a) uchovanie údajov
(b) ochranu údajov
(c) agregáciu údajov
(d) synchronizáciu údajov
(e) uloženie zdrojového kódu

3. (1 b) V jazyku C++ sa preťažením operátora
(a) mení význam tohto operátora zásahom do kompilátora
(b) mení význam tohto operátora pre všetky typy operandov
(c) mení význam tohto operátora pre daný typ operandov
(d) význam tohto operátora vôbec nemení
(e) mení význam tohto operátora zásahom do linkera

4. (1 b) V jazyku C++ sa deštruktor
(a) neuvádza nikdy
(b) musí uviesť v triedach, ktoré sú odvodené od viacerých iných tried
(c) musí uviesť v každej triede
(d) musí uviesť v triedach, ktoré obsahujú virtuálne metódy
(e) musí uviesť v triedach, v ktorých sa pri tvorbe ich objektov dynamicky alokuje pamäť

5. (1 b) V jazyku AspectJ je možné bez úprav jestvujúcich typov
(a) pridať nové metódy, ale nie aj ovplyvniť správanie jestvujúcich metód
(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) plne spravovať vykonanie metódy
(e) pridať nové atribúty, ale nie aj metódy

6. (1 b) V Jave to, že je metóda generická znamená, že
(a) má typ aspoň jedného parametra alebo návratovej hodnoty zastúpený premennou
(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) nemá telo

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

```
class C implements Runnable {
    M m;
    public C(M m) {
        this.m = m;
    }
    public synchronized void run() {
        for (int i = 0; i < 100000; i++) {
            m.b();
            m.a();
        }
    }
}

public class M {
    private char x = 'a', y = 'a';

    public synchronized void a() {
        synchronized(this) {
            if (x != y)
                System.out.println("a");
            x = 'a';
            y = 'a';
        }
    }

    public synchronized void b() {
        if (x != y)
            System.out.println("b");
        x = 'b';
        y = 'b';
    }

    public synchronized void c() {
        if (x != y)
            System.out.println("c");
         x = 'c';
         y = 'c';
    }

    public static void main(String[] args) {
        M m = new M();
        new Thread(new C(m)).start();
        new Thread(new C(m)).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) Composite
(b) Visitor
(c) MVC
(d) Strategy
(e) Observer

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(0);
        new C().g(1);
        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) metóda mení dva atribúty
(b) kód na prepočítanie energie na životy nie je súčasťou zodpovedajúceho prijímača
(c) vnútorná logika sa mieša s používateľským rozhraním
(d) kód na prepočítanie energie na životy nie je súčasťou triedy okna hry
(e) atribúty sú **private** a odvodeným triedam nebudú prístupné

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

```
class C {
    public void f() {
        System.out.print("C");
    }
}
class D extends C {
    public void f() {
      super.f();
        System.out.print("D");
    }
}
class E extends D {
    public void f() {
      super.f();
        System.out.print("E");
    }
}
class F extends E {
    public void f() {
      super.f();
        System.out.print("F");
    }
}
public class M {
    public static void main(String[] args) {
        F o1 = new F();
        D o2 = new E();
        E o3 = new E();
        C o4 = new F();
        C o5 = new D();

        ((E) o1).f();
        System.out.print(" ");

        ((E) o2).f();
        System.out.print(" ");

        ((D) o3).f();
        System.out.print(" ");

        ((C) o4).f();
        System.out.print(" ");

        ((C) o5).f();
    }
}
```

12. (3 b) Trieda, ktorá reprezentuje špeciálny dokument, je odvodená od triedy, ktorá reprezentuje všeobecný dokument. Metóda na zistenie signatára všeobecného dokumentu nezaručuje vrátenie mena signatára, lebo všeobecný dokument nemusí mať signatára. Táto metóda je pri špeciálnom dokumente prekonaná a zaručuje vrátenie mena signatára zo zoznamu povolených signatárov. Týmto sa predpoklady a dôsledky pôvodnej metódy zoslabujú, zosilňujú alebo nemenia? Je týmto dodržaný Liskovej princíp substitúcie (LSP)? Je vhodné od triedy, ktorá reprezentuje všeobecný dokument, odvodiť triedu, ktorá reprezentuje špeciálny dokument?

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

(vyplňte tlačeným písmom)

Priezvisko:

Meno:

13. (10 b) V systéme na správu úloh každá úloha má názov, začiatok realizácie (stačí implementovať ako celé číslo: deň od začiatku projektu), predpokladané trvanie (znovu stačí implementovať ako celé číslo: počet dní od začiatku realizácie) a opis. Úloha môže nadväzovať na (jednu) inú úlohu. Okrem iných, na evidované úlohy sú poskytované tieto dve sumarizácie, ktoré sa automaticky aktualizujú pri zmene údajov úloh:

- zoznam názvov úloh so začiatkom realizácie
- zoznam dvojíc úloh, ktoré sú vo vzťahu nadväznosti

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 prvkov nie je potrebné uvádzať.

V implementácii sa sústreďte na aplikačnú logiku – používateľské rozhranie 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 2021/22
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 20. jún 2022
30

1 d

2 a

3 c

4 e

5 d

6 a

7 C.run(), M.a(), M.c()

8 e

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

10 c

11 CDEF CDE CDE CDEF CD (akceptovaná je aj odpoveď bez medzier)

12 nemenia sa / zosilňujú sa / dodržaný / áno

13 Vhodný je vzor Observer. Predmetom pozorovania mal byť objekt triedy, ktorá slúži na uchovanie úloh (rola Subject), a pozorovateľmi objekty tried, ktoré implementujú sumarizácie (rola Observer).