Objektovo-orientované programovanie 2017/18
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 27. jún 2018

(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) Mechanizmus friend známy z jazyka C++ v Jave

(a) jestvuje pod názvom generics
(b) jestvuje pod rovnakým názvom
(c) jestvuje pod názvom **static**
(d) nejestvuje
(e) jestvuje pod názvom package access

2. (1 b) Java podporuje perzistenciu prostredníctvom

(a) serializácie
(b) agregácie
(c) modularizácie
(d) synchronizácie
(e) radializácie

3. (1 b) Potenciálnou nevýhodou aspektov v jazyku AspectJ je, že

(a) nedokážu ovplyvniť používateľské rozhranie
(b) je možné ich mať iba v malom počte
(c) nedokážu nahradiť jestvujúce metódy v triedach
(d) nedokážu pridať nové prvky do tried
(e) v ovplyvnenom kóde táto skutočnosť nie je viditeľná

4. (1 b) Generickosť v Jave umožňuje, aby zoskupenia

(a) automaticky ukladané na disk
(b) fungovali rýchlejšie
(c) mohli uchovať väčší počet objektov
(d) mohli byť špecializované pre akýkoľvek typ údajov pri inštanciácii
(e) mohli byť špecializované pre akýkoľvek typ údajov pri dedení

5. (1 b) Triedy v jazyku C++ majú pôvod v mechanizme

(a) struct
(b) union
(c) template
(d) define
(e) include

6. (1 b) Tomu, čo sa v Jave implementuje pomocou atribútov s pridruženými prístupovými a nastavovacími metódami (set/get) v jazyku C# zodpovedá mechanizmus

(a) delegate
(b) property
(c) sealed
(d) var
(e) event

7. (2 b) Daná je časť kódu hry v Jave:

```
class Ovladanie {
   public static void main() {
      ...

      if (tah == 'a') { // zautoc na nepriatela
         if (player.hasSword()) {
            enemy.decreaseEnergy(10);
         } else {
            player.decreaseEnergy(10);
         }
      } else if...
      ...
   }
}
```

Z hľadiska flexibility objektovo-orientovaného návrhu by bolo najdôležitejšie

(a) vyčleniť celé ovládanie hry do nestatickej metódy
(b) namiesto príkazu **if** použiť príkaz **case**–**switch**
(c) vyčleniť kód podmienený ťahom `a` do metódy nezávislej od ovládania
(d) zmeniť metódu decreaseEnergy() na statickú
(e) vytvoriť grafické používateľské rozhranie

8. (2 b) Potrebné je zabezpečiť rozšíriteľnosť triedy o ďalšie operácie (metódy), ale aby pri pridávaní operácií nebolo potrebné zasahovať do jej kódu. Ktorý návrhový vzor by ste použili?

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

9. (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.

10. (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 a(char c) throws E {
        if (c == 'a')
            System.out.print("A");
        else
            throw new E();
    }
    public void b(char c) throws E {
        System.out.print("B");
        try {
            a(c);
        } catch (E e) {
            throw e;
        } finally {
            System.out.print("!");
        }
    }
    public static void main(String[] args) throws E {
        new M().b('a');
        new M().b('b');
        new M().b('b');
    }
}
```

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();
        B o3 = new C();
        B o4 = new B();
        A o5 = new B();

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

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

        ((C) o3).x();
        ((C) 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 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 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) 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 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 2017/18
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – opravný termín – 27. jún 2018
30

1 d

2 a

3 e

4 d

5 a

6 b

7 c

8 b

9 A.g()

10 BA!B!

11 AxBxAy CxAyBy CxCy AxBxAyBy AxBxAy

12 nemenia sa / zosilňujú sa / dodržaný

V poslednej otázke mal byť aplikovaný vzor Observer. Predmetom pozorovania mal byť objekt triedy, ktorá slúži na uchovanie úloh (rola Subject), a pozorovateľmi objekty tried, ktoré implementujú sumarizácie (rola Observer).