Objektovo-orientované programovanie **B**

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

Skúška – 9. 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) Java podporuje perzistenciu prostredníctvom

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

2. (1 b) Generickosť v Jave umožňuje, aby zoskupenia

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

3. (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*.

4. (1 b) Tomu, čo sa v Jave implementuje pomocou atribútov s pridruženými prístupovými a nastavovacími metódami (set/get) v jazyku C# zodpovedá mechanizmus

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

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

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

6. (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();
        A o2 = new B();
        C o3 = new C();
        B o4 = new C();
        B o5 = new B();

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

        ((A) o2).x();
        ((A) o2).y();

        System.out.print(" ");

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

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

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

7. (1 b) Mechanizmus friend známy z jazyka C++ v Jave

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

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

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 m(char c) throws E {
        if (c == 'a')
            System.out.print("A");
        else
            throw new E();
    }
    public void p(char c) throws E {
        System.out.print("F");
        try {
            m(c);
        } catch (E e) {
            System.out.print("T");
            throw e;
        } finally {
            System.out.print("E");
        }
    }
    public static void main(String[] args) throws E {
        new M().p('b');
        new M().p('a');
    }
}
```

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

11. (1 b) Potenciálnou nevýhodou aspektov v jazyku AspectJ je, že

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

12. (2 b) Daný je nasledujúci program v Jave:

```
class A {
    private int a;
    private int b;
    private int c;

    public A(int v) {
        b = v;
        c = 2 * b;
    }

    public void x() {
        a--;
        a++;
    }
    public void y() {
        c++;
        c--;
    }
    public void z() {
        c = 2 * b;
    }
    public void w() {
        synchronized(this) {
            if (c != 2 * b)
                System.out.println("!");
        }
    }
}
class B {
    public void m(A o) {
        new Thread() {
            public void run() {
                for (int i = 1; i < 1000000; i++)
                    o.w();
            }
        }.start();

        new Thread() {
            public void run() {
                for (int i = 1; i < 1000000; i++)
                    o.y();
            }
        }.start();

        new Thread() {
            public void run() {
                for (int i = 1; i < 1000000; i++)
                    o.z();
            }
        }.start();

        new Thread() {
            public void run() {
                for (int i = 1; i < 1000000; i++)
                    o.x();
            }
        }.start();
    }
    public static void main(String[] args) {
        A a = new A(9999);
        B b = new B();
        b.m(a);
    }
}
```

Pri ktorých metódach je *nutné* uviesť modifikátor **synchronized**, aby bolo zaručené, že sa výkričník nikdy nevypíše?

(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 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 **B**
doc. Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška – 9. jún 2017
30

1 e

2 b

3 nemenia sa / zosilňujú sa / dodržaný

4 d

5 a

6 AxBxAy AxBxAy CxAyBy CxCy AxBxAyBy

7 c

8 d

9 FTE

10 a

11 e

12 A.y() a A.z(); akceptovaná je aj odpoveď A.y() (k prepnutiu nití pri spracovaní výrazu v metóde A.z() dochádza zriedkavo)

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).

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.