Objektovo-orientované programovanie **B**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška — 26. máj 2009

Priezvisko:

Meno:

1 b	
2 b	
3 b	

Skúška trvá 90 minút.

V otázkach 1–16 je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do tabuľky. Hodnotia sa len odpovede v tabuľke.

V prípade opravy jasne vyznačte odpoveď, ktorá platí. Každá správna odpoveď má hodnotu vyznačenú v otázke. Nesprávna odpoveď, vyznačenie viac odpovedí alebo nejednoznačné vyznačenie má hodnotu 0 bodov. Postup riešenia sa pre otázky 1–16 nehodnotí. Poškodený list nebude uznaný.

Odpovede na otázky 17 a 18 píšte na prídavný list. Na ňom tiež uveďte svoje priezvisko a meno.

	a	b	c	d	e
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

1. (1 b) Ak trieda v jazyku C++ obsahuje čisto virtuálne funkcie, potom
(a) nemôže dediť od triedy, ktorá ich neobsahuje
(b) sa od nej dá dediť len virtuálne
(c) nemôže mať inštancie
(d) môže dediť od inej triedy len virtuálne
(e) nepodporuje polymorfizmus

2. (1 b) Objekt v objektovo-orientovanom programovaní predstavuje
(a) inštanciu triedy
(b) typ
(c) triedu
(d) inštanciu triedy alebo rozhrania
(e) modul

3. (1 b) Atribút triedy, ktorému predchádza kľúčové slovo **protected**
(a) sa nezapíše do súboru pri serializácii objektu
(b) je dostupné len v rámci jednej nite
(c) je dostupné len v danej hierarchii tried
(d) je chránené pred zápisom
(e) je dostupné len v danej triede

4. (1 b) Daný je nasledujúci kód:

```
for (Object o : l) {
    if (o.class == "StrasnyObor")
        ((StrasnyObor)o).utoc();
    else if (o.class == "PlachyObor")
        ((PlachyObor)o).utec();
    else
        ;
}
```

Tento kód porušuje
(a) princíp polymorfizmu
(b) Liskovej princíp substitúcie
(c) princíp generalizácie a špecializácie
(d) princíp otvorenosti a uzavretosti kódu
(e) princíp abstrakcie

5. (2 b) Pri rozhodovaní o použití dedenia je kľúčové sledovať
(a) mieru príbuznosti objektov modelovaných podtypom a nadtypom v realite
(b) možnosti zovšeobecnenia podtypu nadtypom
(c) počet spoločných atribútov a metód podtypu a nadtypu
(d) možnosti abstrakcie podtypu nadtypom
(e) možnosť uplatnenia objektov podtypu namiesto objektov nadtypu

6. (2 b) V rámci Swing sa kliknutie na tlačidlo v okne sleduje
(a) metódou onClick() tlačidla implementovanou pri odvodení od všeobecného tlačidla JButton
(b) prijímačom udalosti kliknutia registrovaným pre dané tlačidlo
(c) automaticky po pridaní tlačidla do okna jeho metódou add()
(d) zavolaním statickej metódy EventQueue.onClick() a následným zistením, či sa kliknutie vzťahuje na dané tlačidlo
(e) volaním metódy actionPerformed() príslušného tlačidla v slučke

7. (2 b) Dedeniu typu extend medzi triedami v Jave v jazyku C++ zodpovedá dedenie typu:
(a) **protected**
(b) **public** s virtuálnymi triedami
(c) **protected** s virtuálnymi triedami
(d) **private**
(e) **public**

8. (2 b) Na rozdiel od objektovo-orientovaného programovania aspektovo-orientované programovanie umožňuje
(a) oddelenie pretínajúcich záležitostí
(b) rýchlejšie vykonávanie programu
(c) rozdelenie kódu do komponentov
(d) prepletenie pretínajúcich záležitostí
(e) tvorbu modulov

9. (2 b) Synchronizácia statickej metódy
(a) znamená uzamknutie objektu **this** pre hocijaký iný prístup
(b) nie je možná
(c) znamená uzamknutie objektu triedy pre iný hocijaký synchronizovaný prístup
(d) znamená uzamknutie objektu **this** pre hocijaký iný synchronizovaný prístup
(e) znamená uzamknutie objektu triedy pre hocijaký iný prístup

10. (2 b) Princíp otvorenosti a uzavretosti hovorí, že
(a) každý prúd údajov okrem štandardného vstupu a výstupu je potrebné po otvorení aj zatvoriť
(b) kód má byť otvorený pre zmeny, ale uzavretý pre rozšírenie
(c) každý prúd údajov je potrebné po otvorení aj zatvoriť
(d) kód má byť otvorený pre rozšírenie, ale uzavretý pre zmeny
(e) kód má byť zároveň aj otvorený, aj uzavretý pre úpravy

11. (2 b) Diagram sekvencií v jazyku UML znázorňuje
(a) štruktúru systému
(b) triedy a vzťahy medzi nimi
(c) funkcionalitu z pohľadu používateľa
(d) vzťahy medzi inštanciami tried v určitom okamihu vykonávania programu
(e) postupnosť volaní medzi objektmi

12. (3 b) V jednej z tried programu sa realizuje výpočet na základe dodaných koeficientov. V prípade ich nesúladu, ktorý sa dá zistiť ich porovnaním, výpočet nemá zmysel. Toto treba ošetriť

(a) vyhodením všeobecnej výnimky typu Exception a jej ošetrením v metóde, ktorá výpočet potrebovala
(b) vyhodením výnimky typu vytvoreného špeciálne pre za týmto účelom a jej ošetrením v metóde, ktorá výpočet potrebovala
(c) vyhodením výnimky typu RuntimeException
(d) priamo v metóde, ktorá realizuje výpočet, pokusom o nápravu výsledku
(e) priamo v metóde, ktorá realizuje výpočet, zastavením programu pre chybu

13. (3 b) Ku kódu v Jave na obr. 1 je daná nasledujúca trieda:

```
class M {
   static void m(Class<? extends X> T, X... o) {
      for (X e : o)
         if (T.isInstance(e))
            System.out.print("i");
   }

   public static void main(String... args) {
      m(X.class, new X[]{new X(), new Y()});
   }
}
```

Jej vykonaním

(a) vznikne výnimka
(b) vypíše sa iii
(c) vypíše sa i
(d) vypíše sa ii
(e) nevypíše sa nič

14. (3 b) Implementácie rozhraní prijímačov vo Swingu predstavujú

(a) návštevníkov vo vzore Visitor
(b) prvky vo vzore Visitor
(c) pozorovateľov vo vzore Observer
(d) kontexty vo vzore Strategy
(e) konkrétne stratégie vo vzore Strategy

15. (3 b) V aspektovo-orientovanej implementácii vzoru Observer

(a) vzniká tretia časť vzoru označovaná ako Mediator medzi časťami Subject a Observer
(b) vyčleňuje sa do aspektu logika vzoru, ktorá je predovšetkým v časti Subject
(c) vyčleňuje sa do aspektu logika vzoru, ktorá je predovšetkým v časti Observer
(d) oddeľuje sa aplikačná logika od rozhrania aplikácie
(e) zaniká potreba implementovať Subject

16. (3 b) Po spustení kódu z obr. 1

(a) vznikne výnimka
(b) vypíše sa yy a vznikne výnimka
(c) vypíše sa xy a program vyhodí výnimku
(d) vypíše sa yyx
(e) vypíše sa yyy

```
class X {
   public void m() { System.out.print("x"); }
}
class Y extends X {
   public void m() { System.out.print("y"); }
}
class A {
   public Runnable r(final X x) {
      return new Runnable(){
         public void run(){
            x.m();
         }
      };
   }
   public static void main(String[] args) {
      X ox = new Y();
      Y oy = new Y();

      Runnable a = (new A()).r(ox);
      Runnable b = (new A()).r(oy);
      Runnable c = (new A()).r((X)oy);
      a.run();
      b.run();
      c.run();
   }
}
```

Obrázok 1: Kód pre otázky 13 a 16.

17. (5 b) Nakreslite diagram tried v UML pre kód na obr. 1 rozšírený o nasledujúcu triedu:

```
class C extends X {
   private X x;
   public void setX(X x) { this.x = x; }
   public X getX() { return x; }
}
```

18. (12 b) Aplikácia umožňuje šifrovanie rôznych druhov súborov. Predpokladajme binárne a textové súbory. Pri textových ešte treba brať do úvahy kódovanie – predpokladajme ASCII a Unicode. Aplikácia poskytuje dva spôsoby šifrovania pre všetky podporené typy súborov. Aplikácia v budúcnosti bude podporovať ďalšie spôsoby šifrovania a možno aj ďalšie druhy súborov.

Napíšte kód v Jave, ktorý implementuje súbory a šifrovanie. Aplikujte pritom mechanizmy objektovo-orientovaného programovania v maximálnej miere a vysvetlite ich úlohu. Samotné spôsoby šifrovania a operácie so súbormi neuvádzajte.

Objektovo-orientované programovanie **B**
Ing. Valentino Vranić, PhD., ÚISI FIIT STU
Skúška — 26. máj 2009

1 c

2 a

3 c

4 d

—

5 e

6 b

7 b

8 a

9 c

10 d

11 e

—

12 b

13 d

14 e

15 b

16 e

50