

Priezvisko:	
Meno:	

1 b	
2 b	
3 b	

Skúška trvá 120 minút.

V otázkach 1–16 je len jedna možnosť správna. Vyznačte svoju odpoveď krížikom do veľkej tabuľky (malú tabuľku nevyplňajte). Hodnotia sa len odpovede vyznačené v tabuľke.

V prípade opravy jasne vyznačte ktorú odpoveď vyberáte. 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.

Odpovede na otázky 17 a 18 píšete 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) V triede, ktorá implementuje rozhranie, je možné zdefinovať

- (a) len ďalšie polia
- (b) ľubovoľné ďalšie polia a metódy
- (c) len ďalšie metódy
- (d) len metódy deklarované v rozhraní
- (e) len metódy nedeklarované v rozhraní

2. (1 b) Ak trieda v jazyku C++ obsahuje čisto virtuálne funkcie, potom

- (a) nemôže dediť od triedy, ktorá ich neobsahuje
- (b) môže dediť od inej triedy len virtuálne
- (c) sa od nej dá dediť len virtuálne
- (d) nepodporuje polymorfizmus
- (e) nemôže mať inštancie

3. (1 b) Objekt v Jave predstavuje

- (a) triedu
- (b) inštanciu triedy alebo rozhrania
- (c) typ
- (d) inštanciu triedy
- (e) modul

4. (1 b) Pole triedy, ktorému predchádza kľúčové slovo **protected**

- (a) je dostupné len v danej hierarchii tried
- (b) je chránené pred zápisom
- (c) je dostupné len v rámci jednej nite
- (d) je dostupné len v danej triede
- (e) sa nezapíše do súboru pri serializácii objektu

5. (2 b) V aspektovo-orientovanej implementácii vzoru Observer

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

6. (2 b) Metóda `f()` triedy `A` vyhadzuje výnimku `MyException`. Daná je trieda `B`:

```
class B {  
    void m() { new A().f(); }  
}
```

Metóda `m()` triedy `B`

- (a) musí deklarovať alebo ošetrovať výnimku typu `MyException`
- (b) je korektná
- (c) musí deklarovať že vyhadzuje výnimku typu `MyException`
- (d) musí ošetrovať výnimku typu `MyException`
- (e) musí vyhadzovať výnimku typu `MyException`

7. (2 b) Jeden z rozdielov medzi abstraktnou triedou a rozhraním v Jave je ten, že

- (a) je možné vytvárať inštancie rozhraní, ale nie aj abstraktných tried
- (b) je možné vytvárať inštancie abstraktných tried, ale nie aj rozhraní
- (c) rozhranie môže dediť od abstraktnej triedy, ale nie naopak
- (d) abstraktná trieda môže dediť od rozhrania, ale nie naopak
- (e) konkrétne triedy môžu dediť od abstraktnej triedy, ale nie od rozhrania

8. (2 b) Program v jazyku C++ obsahuje triedu `Lopta` s implementovanou metódou `skoc()`. Daný je nasledujúci kód:

```
Lopta o;  
o.skoc();
```

Tento kód je v jazyku C++

- (a) korektný, ale jedine ak je metóda `skoc()` statická
- (b) nekorektný
- (c) korektný
- (d) korektný, ale jedine ak je metóda `skoc()` virtuálna
- (e) korektný, ale jedine ak je trieda `Lopta` virtuálna

9. (2 b) Vzťah «use» v jazyku UML v smere od triedy k rozhraniu znamená, že

- (a) trieda ovplyvňuje rozhranie
- (b) trieda implementuje metódy predpísané rozhraním
- (c) trieda volá niektorú z metód rozhrania
- (d) trieda volá všetky metódy rozhrania
- (e) trieda vytvára inštanciu rozhrania

10. (2 b) V jazyku AspectJ je pomocou videnia (advice) možné

- (a) zmeniť vykonávanie metódy
- (b) pridať novú metódu do triedy
- (c) pridať novú metódu do aspektu
- (d) definovať bod spájania
- (e) definovať nové závislosti medzi triedami

11. (3 b) Návrhový vzor Observer je vo vzore Model-View-Controller využitý vo vzťahu

- (a) View-Controller
- (b) Model-Controller
- (c) Model-View
- (d) tried vo vnútri časti View
- (e) tried vo vnútri časti Controller

12. (3 b) Daný je nasledujúci kód v Jave:

```
if (o.class == "Kruh")
    ((Kruh)o).nakresli();
else if (o.class == "Stvorec")
    ((Stvorec)o).nakresli();
else
    ;
```

Tento kód porušuje

- (a) princíp zapuzdrenia
- (b) princíp otvorenosti a uzavretosti
- (c) pravidlá dedenia
- (d) Liskovej princíp substitúcie
- (e) pravidlá polymorfizmu

13. (3 b) Daný je nasledujúci kód v Jave:

```
**1** B {
    **2**
}
```

```
public class A {
    **3** f() {
        return new **3**() {
            public void m() { . . . }
        };
    }
    public void g() {
        new A().f().m();
    }
}
```

Ktoré fragmenty kódu treba v tomto programe doplniť, aby bol korektný?

- (a) \*\*1\*\*: **class**    \*\*2\*\*: **void f();**    \*\*3\*\*: **B**
- (b) \*\*1\*\*: **class**    \*\*2\*\*: **void m();**    \*\*3\*\*: **A**
- (c) \*\*1\*\*: **interface**    \*\*2\*\*: **void f();**    \*\*3\*\*: **B**
- (d) \*\*1\*\*: **interface**    \*\*2\*\*: **void m();**    \*\*3\*\*: **B**
- (e) \*\*1\*\*: **class**    \*\*2\*\*: **void m();**    \*\*3\*\*: **B**

14. (3 b) Ku kódu v Jave na obr. 1 je daná nasledujúca trieda:

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

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

Jej vykonaním

- (a) vypíše sa a
- (b) vypíše sa aa
- (c) vznikne výnimka
- (d) nevypíše sa nič
- (e) vypíše sa aaa

15. (3 b) Daný je kód v Jave na obr. 1. Vykonaním týchto príkazov:

```
A o = new X();
o.m();
((X)o).m();
((A)o).m();
```

- (a) vznikne chyba v poslednom riadku
- (b) pole i nadobudne hodnotu 3
- (c) pole i nadobudne hodnotu 1
- (d) pole i nadobudne hodnotu -1
- (e) pole i nadobudne hodnotu -3

```
abstract class A {
    int i = 0;
    public abstract void m();
}

class X extends A {
    public void m() { i--; }
}

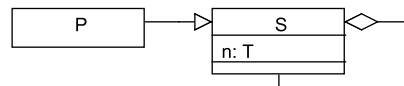
class Y extends X {
    public void m() { i++; }
}
```

Obrázok 1: Kód pre otázky 14 a 15.

16. (3 b) Vytvorili ste tlačidlo t ako komponent rámca Swing a zviditeľnili ste okno, ktoré ho obsahuje. Potrebujete zmeniť označenie (label) tlačidla na text Abc. Urobíte to volaním

- (a) SwingUtilities.invokeLater(new Runnable() { public void run() {t.changeText("Abc");}});
- (b) SwingUtilities.invokeLater(t.changeText("Abc"));
- (c) SwingUtilities.run(new Runnable() { public void invokeLater() {t.changeText("Abc");}});
- (d) t.changeText("Abc")
- (e) (new Runnable(SwingUtilities.invokeLater(t.changeText("Abc");))).run();

17. (4 b) Prepíšte diagram na obr. 2 do kódu v Jave.



Obrázok 2: Diagram pre otázku 17.

18. (12 b) Aplikácia na modelovanie umožňuje vytváranie a spájanie prvkov. Pri vytvorení prvku sa nastaví jeho celočíselná ohodnotenie a počet prvkov, s ktorým ho možno spojiť. Spojenie sa vždy týka dvoch prvkov, je usmernené a pozná ho len zdrojový prvok.

Každý prvok je jedného z troch druhov: Adder, Subtractor alebo Multiplier. Adder zvýši ohodnotenie prvku, s ktorým sa spája, o svoje ohodnotenie. Subtractor zníži ohodnotenie prvku, s ktorým sa spája, o svoje ohodnotenie. Multiplier vynásobí ohodnotenie prvku, s ktorým sa spája, svojim ohodnotením.

Napište relevantný kód v Jave, ktorý zodpovedá týmto požiadavkám, využívajúc pritom mechanizmy objektovo-orientovaného programovania v maximálnej miere. Kód vysvetlite.

1 b

2 e

3 d

4 a

—

5 c

6 a

7 d

8 c

9 c

10 a

—

11 c

12 b

13 d

14 b

15 e

16 a

50