



Priezvisko:

1b	
2b	

Meno:

Test trvá 35 minút.

V otázkach 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 nehodnotí. Len celistvý list bude akceptovaný.

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

1. (2b) K triedam z obr. 1 je daný nasledujúci kód:

```
List<A> list = new ArrayList<A>();  
list.add(new B());  
list.get(0).m();
```

Tento kód

- (a) sa vôbec nepreloží, lebo typ referencie `list` nezodpovedá typu priradeného objektu
- (b) po vykonaní vypíše `b`
- (c) padne počas vykonávania s výnimkou `ClassCastException`
- (d) sa vôbec nepreloží, lebo do zoznamu `list` sa dajú vkladať len objekty typu `A`
- (e) po vykonaní vypíše `a`

```
class A {  
    public void m() { System.out.print("a"); }  
}  
class B extends A {  
    public void m() { System.out.print("b"); }  
}  
class C extends B {  
    public void m() { System.out.print("c"); }  
}  
class M {  
    public static void r(A o) { o.m(); }  
}
```

Obr. 1: Kód pre otázky 1 a 7.

2. (1b) V triede, ktorá dedí od rozhrania je možné zadefinovať

- (a) aj iné metódy než predpisuje rozhranie
- (b) len ďalšie polia
- (c) len to čo predpisuje rozhranie
- (d) len konkrétne metódy
- (e) len metódy ktoré predpisuje rozhranie

3. (1b) Princíp otvorenosti a uzavretosti hovorí, že

- (a) kód je počas úpravy otvorený, a po úprave uzavretý
- (b) kód má byť otvorený pre zmeny, ale uzavretý pre rozšírenie
- (c) súbor s kódom treba po otvorení aj zavrieť
- (d) kód má byť zároveň aj otvorený, aj uzavretý pre úpravy
- (e) kód má byť otvorený pre rozšírenie, ale uzavretý pre zmeny

4. (1b) Agregácia v objektovo-orientovanom programovaní

- (a) predstavuje kritérium pre použitie dedenia
- (b) znamená spájanie objektov do väčších celkov
- (c) umožňuje, aby sa objekt uplatnil namiesto objektu jeho nadtypu
- (d) stanovuje kritéria pre použitie abstraktných tried
- (e) znamená skrytie implementácie objektu

5. (1b) Zapuzdrenie v objektovo-orientovanom programovaní

- (a) umožňuje spájanie objektov
- (b) umožňuje, aby sa objekt uplatnil namiesto objektu jeho nadtypu
- (c) umožňuje znížiť závislosť klientskeho kódu
- (d) predstavuje kritérium pre použitie agregácie
- (e) predstavuje spôsob tvorenia hierarchie

6. (1b) Daná je trieda

```
public class A {  
    private int i;  
    public void m(int i) { this.i = i; }  
}
```

Daný je objekt o triedy `A`. Príkaz

```
o.m(5);
```

- (a) mení stav a správanie objektu
- (b) inicializuje objekt
- (c) mení identitu objektu
- (d) mení stav objektu
- (e) mení správanie objektu

7. (2b) Daný je kód v Jave na obr. 1. Čo sa vypíše po vykonaní týchto príkazov:

```
A a = new A();  
B b = new B();  
A c = new C();
```

```
M.r(a);  
M.r(b);  
M.r(c);  
M.r((A)a);  
M.r((A)b);  
M.r((A)c);
```

- (a) abcaba
- (b) aaaaaa
- (c) abcaaa
- (d) abcabc
- (e) abaaba

8. (1 b) Daný je nasledujúci kód:

```
for (Object o : l) {
    if (o.class == "Frog")
        ((Frog)o).jump();
    else if (o.class == "Bird")
        ((Bird)o).fly();
    else
        ;
}
```

Tento kód porušuje

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

9. (1 b) Daná je trieda:

```
public abstract class T {
    abstract void a();
    void b() { ... };
}
```

Inštancia tejto triedy sa

- (a) dá vytvoriť
- (b) dá vytvoriť, ale nebude sa dať zavolať metóda a()
- (c) nedá vytvoriť
- (d) dá vytvoriť, ale bude ukazovať na **null**
- (e) nedá vytvoriť, lebo metóda a() je abstraktná

10. (1 b) Rozšírená slučka **for** v Jave pri aplikácii na zoskupenie

- (a) potrebuje explicitné zadanie iterátora zoskupenia
- (b) automaticky získa iterátor zoskupenia
- (c) potrebuje explicitné zadanie inkrementátora zoskupenia
- (d) automaticky získa inkrementátor zoskupenia
- (e) pracuje cez iteračnú premennú

11. (1 b) Je dobré z hľadiska objektovo-orientovaného prístupu rozsiahle používanie statických metód?

- (a) Áno, lebo sa dajú prekonať.
- (b) Nie, lebo sa nededia.
- (c) Áno, lebo sa vykonajú rýchlejšie.
- (d) Nie, lebo nepodporujú zapuzdrenie.
- (e) Nie, lebo neumožňujú polymorfizmus.

12. (2 b) Daný je nasledujúci kód:

```
class MyException extends Exception {}

class A {
    void a(int i) throws MyException {
        if (i > 0) {
            i = i / 2;
        }
        else
            throw new MyException();
    }
}
```

```
class B {
    void b(int i) {
        try {
            new A().a(i);
        } catch (MyException e) {
            System.out.println("chyba");
        }
    }
}
```

Metóda b() triedy B

- (a) musí v bloku **try** vyhadzovať výnimku typu **MyException**
- (b) musí v bloku **finally** vyhadzovať výnimku typu **MyException**
- (c) musí v bloku **catch** vyhadzovať výnimku typu **MyException**
- (d) je korektná
- (e) musí obsahovať klauzulu **throws**



15 b

1 b

2 a

3 e

4 b

5 c

6 d

7 d

8 a

9 c

10 b

11 e

12 d