

februar 2020.  
Kvalifikacioni test

K1. Koliko objekata je kreirano tokom izvršavanja narednog koda. Kojim klasama pripadaju?

```
public class InitialTest {  
    int x;  
    public static void main(String[] args) {  
        InitialTest i = new InitialTest();  
        InitialTest i2 = i;  
        Strings s = "Hello!";  
        i.printIt();  
        System.out.println(s);  
    }  
    public void printIt(){  
        int y;  
        y=2;  
        System.out.println(x + " " + y);  
    }  
}
```

1

K2. Šta je od navedenog važi u Java programskom jeziku?

- ☒ A) Interfejs je tip.
- ☐ B) Interfejs nije tip podatka, jer se ne može instancirati.
- ☒ C) Moguće je definisati referencu interfejs tipa.
- ☐ D) Interfejs predstavlja tip podatka, ali primitivan.
- ☐ E) Interfejs se može instancirati pod uslovom da postoji bar jedna klasa koja ga implementira.

K3. Eksplicitan poziv metoda super(...) u konstruktoru dete klase je (u Javi):

- a) obavezno,
- b) nedozvoljeno,
- ☒ c) neophodan potreban ako ne postoji definisan default konstruktor nadklase.

Zaokružiti sva tvrđenja koja se smatraju tačnim.

K4. Da li je moguće moguće definisati reference na sledeći način:

Object o = new String("Hello, world!");

Obrazložiti odgovor.

K5. Oblast važenja imena u Javi.

februar 2020.  
Završni deo ispita – TEST

- Pitanja A kategorije su osnovna i bez davanja tačnih odgovora na njih nije moguće položiti završni deo ispita.
- Na pitanja B kategorije nije obavezno dati odgovor da bi ostala pitanja bila bodovana.

**kategorija A**

- A1. Napisali ste samostalnu Java aplikaciju, uz korišćenje samo standardnih Javinih biblioteka. Aplikacija je smeštena u fajl **Smaug.java** i uspešno je kompajlirali. Šta je minimalno potrebno da nekome dostavite i instalirate da bi na njegovoj/njenoj mašini igra mogla da bude pokrenuta?
- A2. Pojam konstruktora. Konstruktori u Javi.
- A3. Članovi objekata i klasa:
- podaci,
  - metodi.
- A4. Čemu služi ključna reč **super**?
- A5. Paketi u Javi. Koncept, kompajliranje i pokretanje.

**kategorija B**

- B0. UML
- Šta je UML?
  - Apstrakcija/generalizacija/specijalizacija
  - Agregacija/kompozicija
- B1. Šta je rezultat sledećeg koda:
- ```
class Feline {
    public String type = "f ";
    public Feline() {
        System.out.print("feline ");
    }
}
public class Cougar extends Feline {
    public Cougar() {
        System.out.print("cougar ");
    }
}
void go() {
    type = "c ";
    if (this.type == super.type) System.out.print("same ");
    else System.out.print("diff ");
}
public static void main(String[] args) {
    new Cougar().go();
}
}
```

- A) cougar same
- B) cougar diff
- ☒ C) feline cougar same
- D) feline cougar diff
- E) cougar feline same
- F) cougar feline diff

Šta će biti rezultat rada sledećeg programa:

```
class Alpha {
    String getType() { return "alpha"; }
}
class Beta extends Alpha {
    String getType() { return "beta"; }
}
class Gamma extends Beta {
    String getType() { return "gamma"; }
    public static void main(String[] args) {
        Gamma g1 = new Alpha();
        Gamma g2 = new Beta();
        System.out.println(g1.getType() + " "
            + g2.getType());
    }
}
```

- A) alpha beta
- B) beta beta
- C) gamma gamma
- D) alpha alpha
- ☒ E) Compilation fails.

B3. Petoj linij narednog koda dodati komandu kojom će biti kreiran objekat klase **MyInner** vezan za objekat **mt**.

```
1. class TestNested {
2.     public static void main(String[] args) {
3.         String ext = "Access from external class";
4.         MyTopLevel mt = new MyTopLevel();
5.         _____
6.         inner.accessInner(ext);
7.     }
8. }
9. class MyTopLevel {
10.     private String top = "From Top level class";
11.     class MyInner {
12.         public void accessInner(String st) {
13.             System.out.println(st);
14.         }
15.     }
16. }
```

finally block must be executed!

D. A compiler error occurs.

B4. Izmeniti kod tako da nema grešaka u kompajliranju i izvršavanju.

```
1. class AnimalCreator {
2.     public static void main(String [] args) {
3.         Animal [] animals = {new Animal(), new Cow()};
4.         for (Animal a : animals){
5.             Animal x = a.getAnimal();
6.             System.out.println(x);
7.         }
8.     }
9. }
10. class Animal {
11.     Animal getAnimal() {
12.         return new Animal();
13.     }
14. }
15. class Cow extends Animal {
16.     Cow getAnimal () {
17.         return new Cow();
18.     }
19. }
```

Kod nema grešaka.

B5. Realizovati klasu koja implementira stek (LIFO strukturu, last-in-first-out) pomoću dvostruko ulančane liste elemenata tipa String. Klasa treba da omogući sledeće:

- Inicijalizaciju kao prazan stek.
- Inicijalizaciju kopiranjem iz drugog steka.
- Operaciju dodele drugog steka.
- Operaciju dodavanja elementa na stek.
- Operaciju uzimanja elementa sa steka.