

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.

septembar 2019.
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. Objasini sledeće pojmove:
- klasa/objekat
 - implementacija
 - apstrakcija
- A2. Napisati definiciju klase koja sadrži:
- dve privatne promenljive celobrojnog tipa, x i y ;
 - konstruktor koji prima vrednosti na koje setuje x i y ;
 - metod `equals` koji poredi stanje tekućeg objekta sa stanjem objekta koji dobija u argumentu.
- Napisati testnu klasu u kojoj se kreiraju dva različita objekta prethodno definisane klase inicijalizovana istim vrednostima, a zatim ih poredi po stanju, pa po referenci i ispisuje odgovarajuću poruku.
- A3. final modifikator. Objasniti uticaj na podatke i metode. Pokazati opisano na primerima.
- A4. Polimorfizam. Obasnuti pojam i dati primer.
- A5. Kako je u Javi obezbeđena prenosivost koda?

kategorija B

- B0. UML
- Šta je UML?
 - Apstrakcija/generalizacija/specijalizacija
 - Agregacija/kompozicija
- B1. Dati kod ima 2 kompajlerske greške. Koje su i kako ih rešiti?

```
public class Test {  
    int variable = 10;  
    void Test() {  
        variable++;  
    }  
    public int GetVariable() {  
        return variable;  
    }  
    public static void main(String [] args) {  
        Test t = new Test(20);  
        System.out.println(GetVariable());  
    }  
}
```

Handwritten note: t.getVariable();

B2. Dat je deo koda

```
public class Room{  
    class Lab extends Room {  
        _____ // oznaceno mesto  
    }  
}
```

Koji će od navedenih delova koda će ubacivanjem na označeno mesto u polaznom kodu proizvesti kompajlersku grešku?

- A. Lab(){}
- B. Lab(){
 super();
}
- C. Lab(){
 Lab(int i){
 this();
 }
}
- ☒ D. Lab(){
 Lab(int i){
 super();
 this();
 }
}
- ☐ E. Lab(){
 System.out.println("Hello");
 super();
}

B3. Koju liniju koda je moguće smestiti u 7. red a da pri tome kod nema kompajlerskih grešaka?

```
1. interface Animal {  
2.     void saySomething();  
3. }  
4. class Farm {  
5.     void setName(String name){};  
6. }  
7. _____  
8. public class Cow implements Pasture {  
9.     public void graze() { }  
10.    void saySomething(){}11. }
```

☒ A. interface Pasture {void graze();}

B. interface Pasture {void graze();} *Abstract methods do not specify a body.*

C. interface Pasture extends Animal{void graze();} *Cannot reduce the visibility of the inherited method from Animal*

D. interface Pasture extends Animal{void saySomething();}

E. interface Pasture implements Animal{void graze();}

B4. Dati kod

```
abstract class Shape {  
    void draw();  
    void message() {  
        System.out.println("I cannot live without being a parent.");  
    }  
}  
class Circle extends Shape {  
    void draw() {  
        System.out.println("Circle drawn.");  
    }  
}  
class Cone extends Shape {  
    void draw() {  
        System.out.println("Cone drawn.");  
    }  
}
```

```

}
public class RunShape {
    public static void main(String[] args) {
        Object circ = new Circle();
        Cone cone = new Cone();
        Shape[] s = {circ, cone};
        for (Cone inst:s) inst.draw();
        cone.message();
    }
}

```

doraditi tako da da sledeći ispis:

Circle drawn.

Cone drawn.

I cannot live without being a parent.

B5. Definirati klasu koja predstavlja nit koja ispisuje sve parne/neparne brojeve manje od zadatog. Da li će ispisivati parne ili neparne definiše se pozivom konstruktora niti kojem se zadaje 1 u slučaju da treba da ispisuje neparne, a 0 u slučaju da treba da ispisuje parne brojeve. Napisati testnu klasu koja demonstrira rad jedne niti koja ispisuje sve parne brojeve manje od 100 i jedne koja ispisuje sve neparne brojeve manje od 50.