

jul 2020.
Kvalifikacioni test

K1. Šta će biti rezultat izvršavanja narednog koda ukoliko nema kompajlerskih ili grešaka u izvršavanju? Bilo koji scenario da je u pitanju, obrazložiti. Ukoliko postoji problem dati predlog za njegovo rešavanje.

```
public class InitialTest {  
    int x = 5;  
    public static void main(String[] args) {  
        InitialTest a = new InitialTest();  
        InitialTest b = new InitialTest();  
        InitialTest c = b;  
        b = a;  
        a.x++;  
        System.out.println(c.x);  
    }  
}
```

A

B

6
5

K2. Šta je JRE?

K3. Šta će biti rezultat izvršavanja narednog koda ukoliko nema kompajlerskih ili grešaka u izvršavanju? Bilo koji scenario da je u pitanju, obrazložiti. Ukoliko postoji problem dati predlog za njegovo rešavanje.

```
public class Test {  
    public static void main (String[] args) {  
        C c = new C();  
    }  
}  
class A { public A() { System.out.println("Ja sam A"); } }  
class B extends A { public B() { System.out.println("Ja sam B"); } }  
class C extends B { public C() { System.out.println("Ja sam C"); } }
```

K4. Neka je dat sledeći kod:

```
1. public class Test {  
2.     public static void main (String[] args) {  
3.         B b = (B)new A();  
4.         b.jaSam();  
5.     }  
6. }  
7. class A { }  
8. class B extends A {  
9.     public void jaSam() {  
10.         System.out.println("Ja sam B");  
11.     }  
12. }
```

Šta će se desiti pri njegovom pokretanju? (Obrazložiti svaki odgovor)

- A. Greška u kompajliranju linije 3
- B. Greška u kompajliranju linije 4
- C. Greška u kompajliranju linije 7
- D. Biće ispisana poruka Ja sam B
- E. Greška u izvršavanju na liniji 3
- F. Greška u izvršavanju na liniji 4
- G. Ništa od prethodno navedenog.

K5. Šta su podaci članovi klase, a šta podaci članovi objekata?

K*. Šta su automatske promenljive?

008
jul 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. Proveravani i neproveravani izuzeci.
- A2. Interfejsi. Objasniti pojam, način upotrebe. Dati primer.
- A3. Statičko i dinamičko vezivanje.
- A4. Primena modifikatora final. Na šta se može primeniti i šta je efekat njegove primene?
- A5. Na šta se može primeniti synchronized? Šta biva zaključano kada se primeni?

kategorija B

B1. Dati kod ima kompajlerskih grešaka. Koje su i kako ih rešiti? Šta će biti rezultat izvršavanja tako ispravljenog koda?

```
abstract class Animal {  
    abstract public void saySomething() {} ;  
}  
class Goat extends Animal {  
    public void saySomething() {  
        System.out.print(" Mee!");  
    }  
}  
class Cow extends Animal {  
    public void saySomething() {  
        System.out.print(" Moo!");  
    }  
    public void Eat() {  
        System.out.print("Cow eat now.");  
    }  
    public static void main(String [] args) {  
        Animal[] animals = {new Cow(), new Animal(), new Goat()};  
        for(Animal a : animals) {  
            a.saySomething();  
        }  
        animals[0].Eat();  
    }  
}
```

Moo!
Mee!
Cow eat now.

apstraktna klasa
se ne može
instancirati

((Cow)animals[0]).Eat();

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. Dat je kod

```
interface Animal {  
    void saySomething();  
}  
class farm {  
    void setName(String name){};  
}
```

```
public class Cow implements Pasture {  
    public void graze() {}  
    void saySomething(){}
```

Koja od linija dodata na označeno mesto može da omogući prolazak kompajliranja?

- ☒ A. interface Pasture {void graze();}
- B. interface Pasture {void graze(){}}
- C. interface Pasture extends Animal{void graze();}
- D. interface Pasture extends Animal{void saySomething(){}}
- E. interface Pasture implements Animal{void graze();}

B4. Neka su definisana dva tipa niti, od kojih jedna koristi objekat skladišta umanjujući njegovo stanje za 1, a druga dodeljuje uvećava stanje skladišta za 2.

```
class Proizvodjac extends Thread {
    private Skladiste mojeskladiste;
    private int broj;
    private int k;
    public Proizvodjac(Skladiste sk, int broj) {
        mojeskladiste = sk;
        this.broj = broj;
    }
    public void run() {
        for (int i = 0; i < 10; i++) { mojeskladiste.dodaj(); }
    }
}

class Korisnik extends Thread {
    private Skladiste mojeskladiste;
    private int broj;

    public Korisnik(Skladiste sk, int broj) {
        mojeskladiste = sk;
        this.broj = broj;
    }
    public void run() {
        int procitani_broj = 0;
        for (int i = 0; i < 10; i++) procitani_broj = mojeskladiste.uzmi();
    }
}
```

Definisati klasu Skladiste tako da bude thread-safe, onemogući istovremeni pristup/ažuriranje stanja skladišta i obezbedi:

- da stanje skladišta ne može biti negativno
- da se na skladištu ne može naći više od 100 proizvoda

Napisati telo main metoda testne klase u kom se instanciraju po dve niti korisnika i potrošača i startuju.