

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

K2. Šta svi nestatički metodi poseduju, a statički ne? Šta je posledica te razlike.

K3. Da li konstrukcija try/catch/finally mora da sadrži finally blok i u kojim scenarijima se on izvršava.

K4. Neka je dat sledeći kod:

```
1. public class Test{  
2.     public static void main (String[] args) {  
3.         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. Ako klasa poseduje metod run() da li se on može pokenuti kao telo niti?

Avgust 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. this i this()
- A2. Interfejsi.
- A3. Da li se polimorfno ponašanje primenjuje i na statičke metode? Dati i primer.
- A4. Polimorfizam. Obasniti pojam i dati primer.
- A5. Sinhronizacija i nasleđivanje.

kategorija B

B1. Proširiti kod tako da nema ni kompajlerskih, niti grešaka tokom izvršavanja. Šta će biti rezultat izvršavanja tako ispravljenog koda?

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

B2. Da li dati kod ima kompajlerskih ili grešaka u izvršavanju. Ako nema šta je rezultat njegovog izvršavanja? Ako ima, kako ih rešiti?

```
public class Test {
    static final int variable; ← treba inicijalizovati;
    void Test() {
        variable++; ← ne može zbog final
    }
    public int GetVariable() {
        return variable;
    }
    public static void main(String [] args) {
        Test t = new Test();
        System.out.println(variable);
    }
}
```

B3. Dat je kod

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

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

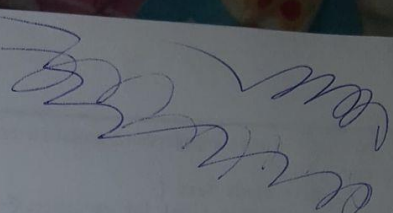
Koja od linija dodata na označeno mesto može da uzrokuje grešku tokom kompajliranja?

- A. interface Pasture {void graze();}
- B. interface Pasture {void graze(){}}
- C. interface Pasture extends Animal{public 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();
}
}

OOP

Definisati klasu Skladiste tako da bude thread-safe, onemogući istovremeni pristup/ažuriranje stanja skladišta i obezbedi da stanje skladišta ne može biti negativno, niti veće od 10.
Napisati telo main metoda testne klase u kom se instanciraju po dve niti korisnika i potrošača i startuju.