

jun 2020.
Kvalifikacioni test

K1. Dopuni main metod tako da bude izvršen metod printIt(), koji za vrednost x ispisuje 5.

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

K2. U prethodni kod dodati this svuda gde se podrazumevano koristi (tako promenjen kod mora uspešno da prođe kompajliranje i da isti izlaz).

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) {  
        B b = new B(5);  
        b.uvecaj(2);  
        System.out.println(b.x);  
    }  
}  
class A {  
    public int x;  
    public A(int vr) { x = vr; }  
    public void uvecaj(int c) { x += c; }  
}  
class B extends A {  
    public B(int vr) { x = vr; }  
    public void uvecaj(int c) {  
        super.uvecaj(c);  
        x *= c;  
    }  
}
```

K4. Neka je dat sledeći kod:

```
1. public class Test{  
2.     public static void main (String[] args) {  
3.         A a = new B();  
4.         a.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. Ništa od prethodno navedenog.

K5. Šta je JVM i čemu služi?

K*. Koja je razlika između deklarisanja i definisanja promenljive?

jun 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. Objasini sledeće pojmove:
enkapsulacija
overloading
- A2. Šta je polimorfizam? Dati primer kada se javlja polimorfno ponašanje. Uslovi za polimorfizam.
- A3. Šta je eksplicitna konverzija i kada ju je potrebno primeniti kada su referencni tipovi u pitanju.
- A4. Anonimne klase.
- A5. Šta su niti? Kako se mogu definisati u Javi korišćenjem interfejsa Runnable? Dati primer.

kategorija B

- B1. Da li je data definicija interfejsa ispravna? Ukoliko nije, napravi izmene tako da je kompajler prihvati. Ukoliko jeste, navedi sve podrazumevane modifikatore koji se primenjuju na ove članove interfejsa.

```
interface merljivo{  
    public static final String jedinica; treba izmisliti konstantu; vrednost  
    public abstract double velicina();  
}
```

- B2. Neka je dat sledeći kod

```
public class Question4{  
    public static void main(String[] args) {  
        int[] x = {0, 1, 2, 3, 4};  
        try{  
            System.out.println("x[5]: " + x[5]);  
            System.out.println("x[3]: " + x[3]);  
        }  
        catch (IndexOutOfBoundsException ie) {  
            System.out.println("Some kind of index out of bound! ");  
        }  
        catch (ArrayIndexOutOfBoundsException oe) {  
            System.out.println("Array index out of bound! " );  
        }  
        finally {  
            System.out.println("finally block must be executed!");  
        }  
        System.out.println("x[0]: " + x[0]);  
    }  
}
```

biti rezultat njegovog kompajliranja i pokretanja? (objasniti)

- A. Array index out of bound!
- B. Some kind of index out of bound!
finally block must be executed!
- C. Some kind of index out of bound!
Array index out of bound!
finally block must be executed!
- D. x[5]: 4
x[3]: 2
finally block must be executed!
- E. x[5]: 4
x[3]: 2

☒ F. Greška u kompajliranju.

H. Ni jedan od ponuđenih odgovora nije tačan.

B3. Šta će biti rezultat izvršavanja sledećeg koda? Ukoliko ima grešaka u kompajliranju i izvršavanju, izmeniti ga.

```
class AnimalCreator {  
    public static void main(String [] args) {  
        Animal [] animals = {new Animal(), new Cow()};  
        for (Animal a : animals){  
            Animal x = a.getAnimal();  
            System.out.println(x);  
        }  
    }  
}  
  
class Animal {  
    Animal getAnimal() {  
        return new Animal();  
    }  
}  
  
class Cow extends Animal {  
    Cow getAnimal () {  
        return new Cow();  
    }  
}
```

*greška. Animal @ Address - objekat
i greška. Cow @ Address - objekat*

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 1.

```
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(); }  
    }  
}
```

OOOP

```
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 prvo se ne može uzimati iz praznog skladišta, ali se može u njega dodavati neograničeno.

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