jun 2019.
Kvalifikacioni test

K1. Dopuni main metod tako da bude izvršen metod printIt(), a zatim ispiši šta će biti rezultat izvršavanja programa.

```
public class InitialTest {
      int x;
   public static void main(String[] args) {


   }
    public void printIt(){
       int y;
       y=2;
     System.out.println(x +" "+ y);
    }
 }
```

K2. Dati primer kako se definiše podatak član klase.

K3. Imajući u vidu naredni kod označi šta je od navedenog tačno (*objasiniti, ukoliko je potrebno*)

```
1. class MyClass {
2.  int i = 5;
3.  static int j = 7;
4.  public static void printSomething () {
5.     i = j;
6.     System.out.println("i: " + i);
7. }
8. }
```

A. Kod se izvršava i daje izlaz
   i: 7
B. Kompajlerska greška na liniji 5.
C. Kompajlerska greška na liniji 6.
D. Greška prilikom izvršavanja.
E. Kod se izvršava i daje izlaz:
   i: 5

***K4.** Neka je data klasa*

```
class Rectangle {
    private int x, y, w, h;
    public boolean equals(Object o){
if(this == o)
        return true;

    }
 }
```

Dopisati telo metoda equals i dati primer upotrebe.

***K5.** Šta je Thread, a šta Runable?*

***K6.** Data je klasa*

```
public class Test { }
```

Koji su dozvoljeni potpisi njenog default-nog konstruktora?

Test( ) / void Test(void) / public Test( ) / public Test(void) / public Test() / public void Test()

jun 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:
enkapsulacija
overloading (preopterećivanje)
polimorfizam

A2. Razlika između base i super.

A3. Šta je eksplicitna konverzija i kada ju je potrebno primentit kada su referencni tipovi u pitanju.

A4. Anonimne klase.

A5. Ako neka je klasa MyNestedClass definisana na sledeći način

```
Consider the following code:
public class MyOuterClass {
        public static class MyNestedClass {
        }
}
```

Na koji od ponuđenih načina je moguće izvesto instanciranje jednog objekta MyNestedClass klase?

A. `MyNestedClass mn = new MyOuterClass.MyNestedClass();`
B. `MyOuterClass.MyNestedClass mn = new MyOuterClass.MyNestedClass();`
C. `MyOuterClass.MyNestedClass mn = new MyNestedClass();`
D. `MyOuterClass mo = new MyOuterClass();`
`MyOuterClass.MyNestedClass mn = mo.new MyNestedClass();`

kategorija B

B0. *UML*

- *Šta je UML?*
- *Apstrakcija/generalizacija/specijalizacija*
- *Agregacija/kompozicija*

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{
    String jedinica;
    double velicina();
}
```

B2. Kakav izlaz daje sledeći kod? Ako ima komajlerskih grešaka ispraviti ih, pa u skladu sa tim dati odgovor na postavljeno pitanje.

```
public class Kartica
{
   public static int brojac = 0;
   private int id;
   private int pin;
   private Racun racun;

   public Kartica(Racun r, int pin) {
         racun = r;
         this.pin = pin;
         id = brojac++;
   }

   public boolean skiniNovacSaRacuna(double iznos)  {
         if(racun.skiniNovacSaRacuna(iznos) != 0.0)
               return true;
         return false;
   }

   public int getPin()  { return pin; }
   public int getId()  { return id; }
   public void setRacun(Racun rac)  { racun=rac; }
   public static void main(String [] args) {
         System.out.println(brojac);
         Kartica k1 = new Kartica();
         System.out.println(k1.brojac);
         System.out.println(k1.getPin());
   }
}
```

Podrazumeva se da je klasa Racun definisana i da ima metod `skiniNovacSaRacuna(double iznos)` i *defult-ni konstruktor.*

B3. *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[6]: " + x[6]);
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]);
}
}
```

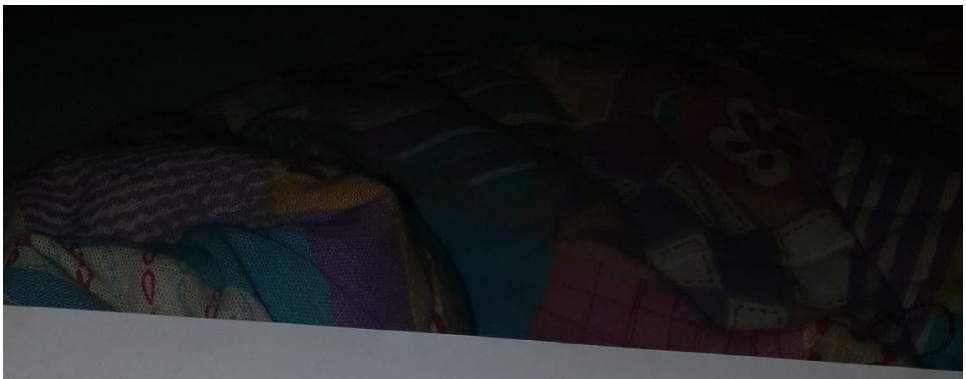
Šta će biti rezultat njegovog kompajliranja i pokretanja?

A. `Array index out of bound!`
   `finally block must be executed!`

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) 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. }
```

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.