1. Uraditi ispis sljedećeg programa:

```
public class A1{
  private A1 a1;
  public A1(){ System.out.println("A1"); }

  public A1(A1 a1){
    System.out.println("A1(a1)");
    this.a1 = a1;
  }

  public static void main(String args[]){
    new A4();
  }

  void metoda(){ System.out.println("metoda A1"); }
}

class A2 extends A1{
  A1 a1;
  public A2(){
    this(new A1());
    System.out.println("A2");
  }

  public A2(A1 a1){
    this.a1 = a1;
    System.out.println("A2(a1)");
  }
}

class A3 extends A2 implements Serializable{
  public A3(){ System.out.println("A3"); }
}

class A4 extends A3{
  private A1 a = new A2();
  private A2 a2 = new A2(new A1(null));
  Serializable a3 = new A3();

  public A4(){
    super();
    System.out.println("A4");
    a.metoda();
  }
}
```

2. Uraditi ispis sljedećeg programa:

```
 public class A {
    static {
    int a = 5;
  System.out.println(a);
 }
 static int a, b;
 public static void main(String args[]) {
   a--;
   metoda();
   System.out.println(a + b + ++a);
   System.out.println(++A.a);
   System.out.println(a);
 }
 public static void metoda() { b = a++ + ++a; }
}
```

3. Uraditi ispis sljedećeg programa:

```
public class B1{
  public void metoda1(){ System.out.println("B1 metoda 1"); }
  public void metoda2(){ System.out.println("B1 metoda 2"); }
  B1(){ System.out.println("B1"); }

  public static void main(String args[]){
    B1 b1 = new B1();
    B1 b2 = new B2(){
      public void metoda1(){ System.out.println("B2 metoda 1"); }
    };

    BI b3 = new B2(){
      public void metoda1(){ System.out.println("B2 m12"); }
      public void metoda2(){ System.out.println("B2 m22"); }
    };
    B1 niz[] = {b1, b2, (B1)b3};
    for(B1 b : niz){
      b.metoda1();
      b.metoda2();
    }
  }
}

abstract class B2 extends B1 implements BI{
  public void metoda2(){ System.out.println("B2 metoda 2"); }
  public B2(){ System.out.println("B2"); }
}

interface BI{
  abstract void metoda1();
  void metoda2(); }
```

4. Uraditi ispis sljedećeg programa:

```
public class C{
  int i = 0;
  public static void main(String args[]){
    C c = new C();
  }

C() {
    while(i < 2 ) {
      System.out.println(i);
      i++;
      continue;
    }
  }
}
```

5. Uraditi ispis sljedećeg programa:

```
class E {
  public static void main(String args[]){
    D d = new D();
    I1 i = new D();
    I2 i2 = d;
    i.metoda();
    d.metodaI3();
    i.metodaI3();
    i2.metoda();
  }
}

interface I1 extends I2, I3 { void metoda(); }
interface I2{ void metoda(); }
interface I3{ void metodaI3(); }

class D implements I1,I3 {
  public void metoda() { System.out.println("metoda"); }
  public void metodaI3(){ System.out.println("metoda I3"); }
}
```

6. Uraditi ispis sljedećeg programa:

```
public class F{
  static int x = 3;
  public static void main(String args[]) { new F(); }

  F(){ this(2); }

  F(int x){
    System.out.println(x);
  }}
```

7. Uraditi ispis sljedećeg programa:

```
public class G1{
  int i;
  {
    System.out.println("blok");
    System.out.println(i = 1);
  }

  protected void test(){
    System.out.println(i);
    System.out.println("iz G1");
  }

  protected void metoda() throws Exception{
    System.out.println("G1 metoda");
    test();
  }

  public static void main(String args[]) throws Exception{
    G1 g1 = new G1(), g2 = new G3();
    g1.metoda();
    g2.metoda();
  }
}

abstract class G2 extends G1{

  abstract protected void metoda() throws Exception;

  protected void test(){
    System.out.println(i);
    System.out.println("iz G2");
  }
}

final class G3 extends G2{

  public void metoda() throws Exception {
    System.out.println("G3 metoda");
    i += 10.51f;
    test();
  }
}
```

8. Uraditi ispis sljedećeg programa (ili naći grešku):

```
public class L extends K{
  public static void main(String args[]){
    L l = new L();
    J j = new J();
    System.out.println(J.i);
    j = (K) l;
    System.out.println(j.metoda());
  }
};

 class J {
  static int i;
  J() { ++i; }
  private int metoda() { return ++i; }
};

class K extends J {
  int i = 0;
  K() { i++; }
  int metoda() { return (i+3); }
};
```

9. Uraditi ispis sljedećeg programa:

```
public class M {
  int x = 0, y = 0;
  M() {}
  M(int a, int b) {
    x = a;
    y = b;
  }

  protected int zbir() { return x+y; }
  protected int razlika() { return x-y; }

  public static void main(String args[]) {
    M m = new M(1,2);
    N n = new N();
    System.out.println(m.razlika());
    System.out.println(n.razlika());
  }
}

class N extends M {
  N() {}
  N(int i, int j) { super(i,j); }
  public int zbir() { return y+x; }
  public int razlika() { return y-x; }
}
```

10. Uraditi ispis sljedećeg programa:

```
public class HelloWorldAnonymousClasses {
  interface HelloWorld {
    public void greet();
    public void greetSomeone(String someone);
  }

  public void sayHello() {
    class EnglishGreeting implements HelloWorld {
      String name = "world";
      public void greet() { greetSomeone("world"); }
      public void greetSomeone(String someone) {
        name = someone;
        System.out.println("Hello " + name);
      }
    }

    HelloWorld englishGreeting = new EnglishGreeting();

    HelloWorld frenchGreeting = new HelloWorld() {
      String name = "tout le monde";
      public void greet() { greetSomeone("tout le monde"); }
      public void greetSomeone(String someone) {
        name = someone;
        System.out.println("Salut " + name);
        ispis();
      }
      public void ispis() { System.out.println("Dodao"); }
    };

    HelloWorld spanishGreeting = new HelloWorld() {
      String name = "mundo";
      public void greet() { greetSomeone("mundo"); }
      public void greetSomeone(String someone) {
        name = someone;
        System.out.println("Hola, " + name);
      }
    };
    englishGreeting.greet();
    frenchGreeting.greetSomeone("Fred");
    spanishGreeting.greet();
  }

  public static void main(String... args) {
    HelloWorldAnonymousClasses myApp =
      new HelloWorldAnonymousClasses();
    myApp.sayHello();
  }
}
```

11. Napisati izlaz sljedećeg programa:

```
class A1 extends A2{
  public static void main(String args[]){
    A1 a1 = new A1();
    A2 a2 = new A2();
    a1.metoda();
  }
 public void metoda(){ super.metoda(); }
}

class A2 extends A3{
  public A2(){ System.out.println("A2()"); }
  public void metoda(){
   this.metoda();
   super.metoda();
   System.out.println(a++);
  }
}

class A3{
  double a;
  int b;
  float c;
  public A3(){
    System.out.println("A3()");
    a = c = b = 1;
  }
  public void metoda(){ System.out.println(a + b++); }
}
```

12. Uraditi ispis sljedećeg programa:

```
public abstract class B1{
  B1(){
    super();
    System.out.println("B1()");
  }

 public abstract void redefinisi(){}
 public static void main(String args[]){
    B3 b3 = new B3();
    b3.metoda();
    B2 b2 = b3;
    b2.metoda();
    B1 b1 = b2;
    b1.metoda();
  }

  private void metoda(){ System.out.println("B1 metoda"); }
}
```

```
abstract class B2 extends B1{
  B2(){ System.out.println("B2()"); }
  abstract protected void metoda();
  void metoda2(){ System.out.println("B2 metoda"); }
}

final class B3 extends B2{
  B3(){
    super();
    System.out.println("B3()");
  }
  public void metoda(){ System.out.println("B3 metoda"); }
}
```

13. Uraditi ispis sljedećeg programa(za slucaj kada C2 nasljeđuje C1 i slučaj kada C3 nasljeđuje C2 i kada nijedna od pomenutih klasa ne nasljeđuje navedenu):

```
public class C1{

  C1(){ System.out.println("C1()"); }
  public static void main(String args[]) throws IOException{
    C1 c1 = new C1();
    try{
      c1.metoda();
      System.out.println("main 1");
    }catch(CE2 e){
      System.out.println("main 2:"+e);
    }catch(CE1 e){
      System.out.println("main 3:"+e);
    }catch(Throwable e){
      System.out.println("main 4:"+e);
    } finally {
     System.out.println("Finally iz maina");
    }
    try(BufferedReader bf = new BufferedReader(new
FileReader("proba.txt"));)
     {
       throw new IOException();
     }

  }
  void metoda() throws Throwable{
    C2 c2 = new C2();
    try{
      c2.metoda();
      System.out.println("C1 : metoda()");
    }finally{
      System.out.println("finally");
    }
```

```
      }
    }

    class C2 /*extends C1*/ {
      C2(){
        System.out.println("C2()");
      }
      void metoda()throws CE1 {
        C3 c3 = new C3();
        System.out.println("C2 : metoda()");
        c3.metoda();
      }
    }
    class C3 /*extends C2*/{
      C3(){
        System.out.println("C3()");
      }
      protected void metoda()throws CE1{
        System.out.println("C3 : metoda()");
        throw new CE2("CCCCEEEE2");
      }
    }
    class CE1 extends Throwable{
      CE1(String s){
        super(s);
        System.out.println("CE1:"+s);
      }
    }
    class CE2 extends CE1{
      CE2(String s){
        super(s);
        System.out.println("ce2:"+s);
      }
    }
```

14. Uraditi ispis sljedećeg programa:

```
    public class D1 extends D3 implements DI{

      public static void main(String args[]){
        D3 niz[]={new D3(), new D2(), new D1()};
        for(int i = 0; i < niz.length; i++){
          niz[i].metoda();
        }
      }
      public D1 metoda(){
        System.out.println("D1: metoda()");
        return (D1) super.metoda();
      }
    }
```

```
class D2 extends D3{
  public D2 metoda(){
    System.out.println("D2: metoda()");
    return new D2();
  }
}

class D3{
  public D3 metoda(){
    System.out.println("D3: metoda()");
    return new D3();
  }
}

interface DI{ D3 metoda(); }
```

15. Nacrtati nit i ilustrovati kako ona radi.

```
public class E1{
  static public void main(String args[]){
    System.out.println("main 1");
    E3 e3 = new E3();
    E2 e2 = new E2(e3);
    e2.start();
    System.out.println("main 2");
  }
}

class E2 extends Thread{
  E3 e3;
  public E2(E3 e3){
    this.e3 = e3;
    System.out.println("E2");
  }
  public void run(){
    for(int i = 0; i < 6; i++)
      System.out.println("E2 run");
  }
  public synchronized void start(){
    super.start();
    new Thread(e3).start();
  }
}

class E3 implements Runnable{
  public E3(){ System.out.println("E3"); }
  public void run(){
    for(int i = 0; i < 6; i++)
      System.out.println("E3 run");
  }
}
```

16. Napisati izlaz sljedećeg programa:

```
public class A1{

  private A1 a1;

  public A1(){ System.out.println("A1"); }

  public A1(A1 a1){
    System.out.println("A1(a1)");
    this.a1=a1;
}

  public static void main(String args[]){
    new A4();
  }

  void metoda(){ System.out.println("metoda A1"); }
}

class A2 extends A1{
  A1 a1;

  public A2(){
    this(new A1());
    System.out.println("A2");
  }

  public A2(A1 a1){
    this.a1=a1;
    System.out.println("A2(a1)"); }
}

class A3 extends A2 implements Serializable{
  public A3(){ System.out.println("A3"); }
}

class A4 extends A3{

  private A1 a = new A2();
  private A2 a2 = new A2(new A1(null));
  Serializable a3 = new A3();

  public A4(){
    super();
    System.out.println("A4");
    a.metoda();
  }
}
```

17. Napisati izlaz sljedećeg programa:

```
public class C1{
  public static void main(String args[]){
    C1 c1 = new C1();
    C2 c2 = new C2();
    try{
      System.out.println(c1.metoda(c1));
      System.out.println(c1.metoda(c2));
      System.out.println(c2.metoda(c1));
      System.out.println(c2.metoda(c2));
    }catch(CE1 e){
      System.out.println("exception 1");
    }finally{
      System.out.println("finally");
    }
  }

  Object metoda(C1 c) throws CE1{
    if(c instanceof C1){
      System.out.println("metoda");
    }else{
      throw new CE2();
    }
    return 1;
  }
}

class C2 extends C1{
  Object metoda (C1 c) throws CE1{
    if(errorCheck()&& c instanceof C2){
      throw new CE2("error 2");
    }else if(c instanceof C2){
      throw new CE1();
    }else{
      return new String("abc");
    }
  }
  boolean errorCheck(){ return true; }
}

class CE1 extends Throwable{
  public CE1(){ System.out.println("ce1 - 1"); }
  public CE1(String s){
    super(s);
    System.out.println("ce1 - 2");
  }
}
```

```
class CE2 extends RuntimeException {
  public CE2(){
System.out.println("ce2 - 1");
}

public CE2(String s){
    this();
    System.out.println("ce2 - 2");
  }
}
```

18. Nacrtati nit i ilustrovati kako ona radi.

```
public class E1 extends Thread{

  private String name;

  public E1(String name){
      this.name = name;
}

  public void run(){
    Runnable r = new Runnable(){
      public void run(){
        for(int i = 0; i < 5; i++){
          System.out.println(i);
        }
      }
    };

    new Thread(r).start();
    synchronized(this){
      for(int i = 0; i < 100; i++){
        System.out.println(i + " " + this.name);
      }
    }
  }

  public static void main(String args[]){
    E1 a = new E1("A");
    E1 b = new E1("B");
    a.start();
    b.start();
  }
}
```

19. Napisati izlaz sljedećeg programa:

```
public class D1 extends DI1.D2 implements DI1,DI2{
  public D1(){
    super();
    System.out.println("D1()");
  }

  public static void main(String args[]){
    DI2 di1 = new D1();
    DI2 di2 = new D2();
    DI1 di3 = new D1();
    D1 d1 = new D1();
    D2 d2 = new DI1.D2();
    System.out.println(((DI2)new D1()).metoda());
    System.out.println(((DI2)d2).metoda());
  }

//public int metoda(){
public void metoda(int i){
    System.out.println("D1 metoda");
    return 1;
  }
}

interface DI1{
  class D2 implements DI2{

    D2(){ System.out.println("D2()"); }
    public int metoda(){
      System.out.println("D2 metoda");
      return 0;
    }
  }
}

interface DI2{ int metoda(); }
```

Šta će se desiti kada odkomentarišemo public int metoda() i return 1, a zakomentarišemo public void metoda(int i)?

20. Napisati izlaz sljedećeg programa:

```
public class F1{

  static boolean b;
  static int counter = 0;

  public static void main(String args[]){

    try{
      label:
        while(counter == 0){
        if(!b){
          System.out.println("C");
          b = true;
          main(new String[]{"D"});
          System.out.println("E");
        }else{
          counter++;
          System.out.println("A");
          System.out.println(args[0]);
          main((String[])new Object("F"));
          if(counter == 1){
            continue label;
            System.out.println("B");
          }
        }
        System.out.println(F1.getF1());
      }

    }catch(Exception e){
      System.out.println("exception");
    }
  }

  public static F1 getF1(){
    System.out.println("1");
    return new F1();
  }

  {
    System.out.println("F1");
  }
}
```

Ukoliko postoji greska napisati gdje se nalazi, pa ispisati izlaz programa u slučaju da se zakomentariše linija sa greškom.

21. Napisati izlaz sljedećeg programa:

```
public class G1{
  public static void main(String args[]) throws Exception{
    G2 g2 = new G2();
    G3 g3 = new G3("a");
    ObjectOutputStream cout = new ObjectOutputStream(new
FileOutputStream("G1.out"));
    cout.writeObject(g2);
    cout.writeObject(g3);
    ObjectInputStream cin = new ObjectInputStream(new
FileInputStream("G1.out"));
    G2 g22 = (G2)cin.readObject();
    System.out.println(g22.a);
    System.out.println(g22.b);
    G3 g33 = (G3)cin.readObject();
    System.out.println(g33.a);
    System.out.println(g33.b);
    cin.close();
  }
}
class G2 implements Externalizable{
  int a = 1;
  transient int b = 2;
  public G2(){ System.out.println("G2 konstruktor"); }
  public void writeExternal(ObjectOutput out)throws IOException{
    out.write(3);
    out.write(4);
    System.out.println("G2 writeExternal");
  }
  public void readExternal(ObjectInput in)throws
IOException,ClassNotFoundException{
    System.out.println("G2 readExternal");
  }
}
class G3 implements Serializable{
  int a = 5;
  transient int b = 6;
  public G3(String s){ System.out.println("G3 konstruktor"); }
  private void writeObject(ObjectOutputStream out)throws
IOException{
    System.out.println("G3 writeObject");
    out.write(a);
    out.write(b);
  }
  private void readObject(ObjectInputStream in)throws
IOException, ClassNotFoundException{
    System.out.println("G3 readObject");
    a = in.read();
    b = in.read();
  } }
```

22. Napisati izlaz sljedećeg programa:

```
public class A {
  public static void main(String[] args) {
    String a = "newspaper";
    a = a.substring(5,7);
    System.out.println(a);
    char b = a.charAt(1);
    System.out.println(b);
    a = a + b;
    System.out.println(a);
  }
}
```

23. Napisati izlaz sljedećeg programa:

```
public class A{
  public static void main(String[] args) {
    String a = "newspaper";
    a = a.substring(5,7);
    System.out.println(a);
    char b = a.charAt(1);
    System.out.println(b);
    a = a + b;
    System.out.println(a);
    C c = new C();
    c.ispis();
    c.metoda();
  }
}
abstract class B extends A {
  protected abstract void metoda();
  protected String ispis(){ return "ispis"; }
}
class C extends B{
  private void metoda(){ System.out.println("C"); }
}
```

24. Uraditi ispis sljedećeg programa:

```
public class AnonimnaKlasa {
    public static void main(String args[]){
        int i = 1, j = 0;
        switch(i){
            case 2: j += 6;
            case 4: j += 1;
            default: j += 2;
            case 0: j += 4;
        }
        System.out.println("j = " + j);
    } }
```

25.Uraditi ispis sljedećeg programa:

```
public class Animal {

  public void eat(){ System.out.println("Animal is eating"); }

  public void drink(){ System.out.println("Animal is drinking");}

  private void privateMethod() {
    System.out.println("Animal's private method");
  }

  public void commonMethod() { System.out.println("Common
method"); }

  public static void main(String[] args) {

    Animal a = new Cat();
    a.eat();
    a.drink();
    a.privateMethod();
    a.commonMethod();
    ((Cat)a).meow();

    Cat c = (Cat)a;
    c.eat();
    c.drink();
    ((Animal)c).privateMethod();
    c.commonMethod();
    c. meow();

    Animal animal = new Animal();
    Cat catty = (Cat) animal;
  }
}

class Cat extends Animal {

  public void eat() { System.out.println("Cat is eating"); }

  public void drink() { System.out.println("Cat is drinking"); }

  public void meow() { System.out.println("Cat is meowing"); }
}
```

26. Napisati izlaz sljedećeg programa:

```
public abstract class B1 {

  B1() {
    super();
    System.out.println("B1()");
  }

  public static void main(String[] args) {
    B3 b3 = new B3();
    b3.metoda();
    B2 b2 = new B2();
    b2 = b3;
    b2.metoda();
    B1 b1 = b2;
    b1.metoda();
    B2 test2 = new B2();
    test2.metoda();
  }

   void metoda() { System.out.println("B1 metoda..."); }

}

 class B2 extends B1 {
  B2() { System.out.println("B2()"); }

  protected void metoda() { System.out.println("B2 metoda..."); }

  void metoda2() { System.out.println("B2 metoda..."); }
}

final class B3 extends B2 {

  B3() {
    super();
    System.out.println("B3()");
  }

  public void metoda() {
    System.out.println("B3 metoda...");
  }
}
```

27. Napisati izlaz sljedećeg programa:

```
class BaseClass {
  private void foo(){ System.out.println("In BaseClass.foo()"); }
  void bar(){ System.out.println("In BaseClass.bar()"); }

  public static void main(String[] args) {
    DerivedClass po = new DerivedClass();
    ((BaseClass)po).foo();
    ((BaseClass)po).bar();
  }
}

class DerivedClass extends BaseClass {
  void foo(){ System.out.println("In Derived.foo()"); }
  void bar(){ System.out.println("In Derived.bar()"); }
}
```

28. Napisati izlaz sljedećeg programa:

```
private class KlasaA {

  public static void main(String ... args){
    int a = 0;
    int b = 5;
    System.out.println(a++ + b);
  }
}
```

29. Napisati izlaz sljedećeg programa:

```
class Bonds {
   Bonds force() { return new Bonds(); }
 }
 public class Covalent extends Bonds {

   Covalent force() { return new Covalent(); }

   public static void main(String[] args) {
     new Covalent().go(new Covalent());
   }

   void go(Covalent c) {
     go2(new Bonds().force(), c.force());
   }

   void go2(Bonds b, Covalent c) {
     Covalent c2 =(Covalent)b;
     Bonds b2 = (Bonds)c;
   }
 }
```

30. Napisati izlaz sljedećeg programa:

```
public class Cycles {

  public static void rideCycle(CycleFactory factory) {
    Cycle c = factory.getCycle();
    c.ride();
  }

  public static void main(String [] args) {
    rideCycle(new UnicycleFactory());
    rideCycle(new BicycleFactory());
    rideCycle(new TricycleFactory());
  }
}

interface Cycle { void ride(); }

interface CycleFactory { Cycle getCycle(); }

class Unicycle implements Cycle {
 public void ride() {
   System.out.println("Ride Unicycle");
 }
}

class UnicycleFactory implements CycleFactory {
 public Cycle getCycle() {
  return new Unicycle();
 }
}

class Bicycle implements Cycle {
 public void ride() { System.out.println("Ride Bicycle"); }
}

class BicycleFactory implements CycleFactory {
 public Cycle getCycle() { return new Bicycle(); }
}

class Tricycle implements Cycle {
  Tricycle() { System.out.println("Tricycle()"); }
  public void ride() { System.out.println("Ride Tricycle"); }
}

class TricycleFactory implements CycleFactory {
  public Cycle getCycle() {
    return new Tricycle();
  }
}
```

31. Napisati izlaz sljeđećeg programa:

```
class Equals {
  public static void main(String [] args){
        int x = 100;
        double y = 100.1;
        boolean b = (x = y);
        System.out.println(b);
    }
}
```

32. Napisati izlaz sljedećeg programa:

```
class EqualsS{
    public static void main(String [] args) {
        int x = 97;
        char y = 'a';
        boolean b = (x == y);
        System.out.println(b);
    }
}
```

33. Napisati izaz sljedećeg programa:

```
public class Ex11 {
  public static void main(String[] args) {
    Test t = new Test();
    t.f().say("hi");
    ((Inner)t.f()).say("hello");
  }
}

interface Ex11Interface {
  void say(String s);
}

class Test {
  private class Inner implements Ex11Interface {
    public void say(String s) {
      System.out.println(s);
    }
  }
  Ex11Interface f() {
    return new Inner();
  }
}
```

34. Napisati izlaz sljedećeg programa:

```
public class Exercise7 {
  private int x;
  private void metoda() {
    System.out.println("Exercise7.metoda()");
  }

  private void metoda2() {
    Exercise7Inner e7i = new Exercise7Inner();
    e7i.metodaModify();
    System.out.println(x);
    System.out.println(e7i.y);
  }

  private class Exercise7Inner {
    private int y;
    private void metodaModify() { x = 5; }
  }

  public static void main(String[] args) {
    Exercise7 e7 = new Exercise7();
    e7.metoda2();
  }
}
```

35. Napisati izlaz sljedećeg programa:

```
public class Testiranje{
    public static void main(String args[]){
        try {
            throw new Exc1();
        }catch (Exc0 e0){
            System.out.println("Ex0 caught");
        }catch (Exception e){
            System.out.println("exception caught");
        }
    }
}
class Exc0 extends Exception { }
class Exc1 extends Exc0 { }
```

36. Napisati izlaz sljedećeg programa:

```
public class G{
  static int x = 3;
  public static void main(String args[]) { new G(); }
  G(){ new G(2); }
  G(int x){ System.out.println(x); }
}
```

37. Napisati izlaz sljedećeg programa:

```
public class ExamQuestion7{
    static int j;
    static void methodA(int i){
        boolean b;
        do{
            b = i<10 | methodB(4);
            b = i<10 || methodB(8);
        }while (!b);
    }
    static boolean methodB(int i){
        j += i;
        return true;
    }
    public static void main(String[] args){
        methodA(0);
        System.out.println( "j = " + j );
    }
}
```

38. Napisati izlaz sljedećeg programa:

```
public class Go extends Game {
  Go() {  super(s2); }
  {
    s += "i ";
  }
public static void main(String[] args) {
    new Go();
    System.out.println(s);
  }
static { s += "sb "; }
}

class Game {
  static String s = "-";
  String s2 = "s2";
  Game(String arg) { s += arg; }
}
```

39. Napisati izlaz sljedećeg programa:

```
public class Igra {
  public static void main(String[] args) {
    Integer i = new Integer(5);
    Integer j = new Integer(5);
    System.out.println(i == j);
    System.out.println( i == (new Integer(5))); } }
```

40.Napisati izlaz sljedećeg programa:

```
public class HummingBird extends Bird {

   public static void fly() { s += "hover "; }

   public static void main(String[] args) {
     Bird b1 = new Bird();
     Bird b2 = new HummingBird();
     Bird b3 = (HummingBird)b2;
     HummingBird b4 = (HummingBird)b2;

     b1.fly(); b2.fly(); b3.fly(); b4.fly();
     System.out.println(s);
   }
 }

 class Bird {
   public static String s = "";
   public static void fly() { s += "fly "; }
}
```

41. Napisati izlaz sljedećeg programa:

```
public class Izuzetak{
  public static void main(String[] args) {
    try{
      int x = 0;
      int y = 5 / x;
    }
    catch (Exception e){
      System.out.println("Exception");
    }catch (ArithmeticException ae) {
      System.out.println(" Arithmetic Exception");
    }
    System.out.println("finished");
  }
}
```

42. Napisati izlaz sljedećeg programa:

```
class MaskiranjeClanova {
  int x = 5;
}
public class MaskiranjeClanovaDva extends MaskiranjeClanova {
  int x = 6;
  public static void main(String[] args) {
    MaskiranjeClanova mc = new MaskiranjeClanova();
    MaskiranjeClanova mcd = new MaskiranjeClanovaDva();
    System.out.println(mcd.x + " " + mc.x);
  }}
```

43.Napisati izlaz sljedećeg programa:

```
interface Counter {
int next();
}

public class LocalInnerClass {
  private int count = 0;

  Counter getCounter(final String name){

    class LocalCounter implements Counter {
      public LocalCounter(){
        System.out.println("LocalCounter()");
      }
      public int next(){
        System.out.println(name);
        return count++;
      }
    }
    return new LocalCounter();
  }

  Counter getCounter2(final String name) {
    return new Counter() {
      {
        System.out.println("Counter()");
      }

      public int next() {
        System.out.println(name);
        return count++;
      }
    };
  }
  public static void main(String[] args) {
    LocalInnerClass lic = new LocalInnerClass();
    Counter
      c1 = lic.getCounter("Local inner "),
      c2 = lic.getCounter2("Anonymous inner ");
    for(int i = 0; i < 5; i++)
      System.out.println(c1.next());
    for(int i = 0; i < 5; i++)
      System.out.println(c2.next());
  }
}
```

44. Napisati izlaz sljedećeg programa:

```
public class MyExceptionTest {

  public static void main(String[] args) {
    try {
      second();
    }catch(MyException e) {
      System.out.println("Uhvacen i obradjen izuzetak " + e);
    }
  }

  public static void second() throws MyException { first(); }

  public static void first() throws MyException {
    throw new MyException("Poruka o gresci");
  }
}

class MyException extends Exception {
  MyException() {
    super();
  }

  MyException(String s) {
    super(s);
    System.out.println("MyException super");
  }
}
```

45. Nacrtati nit i ilustrovati kako ona radi.

```
class Test {
  public static void main(String []args) {
    new MyThread("nit");
  }
}

class MyThread extends Thread {

  public MyThread(String name) {
    this.setName(name);
    start();
    System.out.println("MyThread " + getName());
  }
  public void start() {
    System.out.println("start " + getName());
  }
  public void run() {
    System.out.println("run " + getName());
  }}
```

46. Nacrtati nit i ilustrovati kako ona radi.

```
class MyThread extends Thread {
  public void run(){
    System.out.println("run - ime niti: " +
Thread.currentThread().getName());
  }
  public static void main(String args[])throws Exception{
    Thread myThread = new MyThread();
    myThread.start();
    MyThread nit = new MyThread();
    nit.start();
    nit.join();
    System.out.println("main - ime niti: " +
Thread.currentThread().getName());
  }
}
```

47. Napisati izlaz sljedećeg programa:

```
class One {
  void go1() { System.out.print("1 "); }
  final void go2() { System.out.print("2 "); }
  private void go3() { System.out.print("3 "); }
}

public class OneB extends One {
  void go1() { System.out.print("1b "); }
  void go3() { System.out.print("3b "); }

  public static void main(String[] args) {
    new OneB().go1();
    new One().go1();
    new OneB().go2();
    new OneB().go3();
    new One().go3();
 }
}
```

48. Uraditi izlaz sljedećeg programa:

```
class PassA {
    public static void main(String [] args) {
        PassA p = new PassA();
        p.start();
    }
    void start() {
        long [] a1 = {3,4,5};
        fix(a1);
        System.out.println(a1[0] + " " + a1[1] + " " + a1[2]);
    }
    long [] fix(long [] a3) {
        a3[1] = 7;
        return a3;
    }
}
```

49. Napisati izlaz sljedećeg programa:

```
public class PenguinTest {
  public static void main(String []args) {
    Penguin pingu = new Penguin();
    pingu.walk();
    pingu.fly();
  }
}
class CannotFlyException extends Exception {}

interface Birdie {
  public abstract void fly() throws CannotFlyException;
}

interface Biped { void walk(); }

abstract class NonFlyer {
  private void fly() { System.out.print("cannot fly "); }
}

class Penguin extends NonFlyer implements Birdie, Biped {
  public void walk() { System.out.print("walk\n"); }
}
```

50. Napisati izlaz sljedećeg programa:

```
  public class Players {
  public static void main(String[] args) throws IOException,
ClassNotFoundException {
    System.out.println("Constructing objects:");
    Player1 b1 = new Player1();
    Player2 b2 = new Player2();
    ObjectOutputStream o = new ObjectOutputStream( new
FileOutputStream("Players.out"));
    System.out.println("Saving objects:");
    o.writeObject(b1);
    o.writeObject(b2);
    o.close();
    ObjectInputStream in = new ObjectInputStream( new
FileInputStream("Players.out"));
    System.out.println("Recovering b1:");
    b1 = (Player1)in.readObject();
    System.out.println("Recovering b2:");
    b2 = (Player2)in.readObject();
  }
}

class Player1 implements Serializable {
  public Player1() {
    System.out.println("Player1 Constructor");
  }
   public void writeExternal(ObjectOutput out)
      throws IOException {
    System.out.println("Player1.writeExternal");
  }
  public void readExternal(ObjectInput in) throws IOException,
ClassNotFoundException {
    System.out.println("Player1.readExternal");
  }
}

class Player2 implements Externalizable {
  public Player2() {
    System.out.println("Player2 Constructor");
  }
  public void writeExternal(ObjectOutput out) throws IOException
{
    System.out.println("Player2.writeExternal");
  }
  public void readExternal(ObjectInput in) throws IOException,
ClassNotFoundException {
    System.out.println("Player2.readExternal");
  }
}
```

51. Napisati izlaz sljedećeg prgrama:

```
public class A1{
  static { System.out.println("staticki A1"); }
  private A1 a1;
  public A1(){ System.out.println("A1"); }

  public A1(A1 a1){
    System.out.println("A1(a1)");
    this.a1 = a1;
  }
  public static void main(String args[]){ new A4(); }
  public void metoda(){ System.out.println("metoda A1"); }
}

class A2 extends A1{
  static { System.out.println("staticki A2"); }
  public A1 d = new A1(null);
  A1 a1;
  public A2(){
    this(new A1());
    System.out.println("A2");
  }

  public A2(A1 a1){
    this.a1 = a1;
    System.out.println("A2(a1)");
  }
  static A3 as = new A3();
}

class A3 extends A2 implements Serializable{
  static { System.out.println("staticki A3"); }
  public A1 adas = new A1(null);
  static A1 as = new A1(null);
  public A3(){ System.out.println("A3"); }
}

class A4 extends A3{
  private A1 a = new A2();
  {
    System.out.println("nestaticki A4");
  }
  static A2 asd = new A2(null);
  private A2 a2 = new A2(new A1(null));
  Serializable a3 = new A3();
  public A4(){
    super();
    System.out.println("A4");
    a.metoda();
  }}
```