1. Analizirati kod u primjerima. Ako se program ne?e uspješno kompajlirati, podvu?i problemati?ne linije. Samo ako ?e se kod uspješno kompajlirati, napisati ispis programa. Ako ?e biti ba?en izuzetak, nazna?iti to u ispisu.

SAVJET: Ukoliko se neki blok ispisa ponavlja više puta dovoljno je jednom napisati taj blok, jedinstveno ga ozna?iti, te koristiti tu istu oznaku pri daljnjem pisanju ispisa programa, na mjestima gdje se pri ispisu taj blok ponavlja.

SAVJET: Iako ?e na usmenom biti dovoljno prona?i jednu kompajlersku grešku, ukoliko se desi kompajlerska greška, radi vježbe, analizirati kod kakav bi bio ako bi se ta greška prepravila na proizvoljan na?in (npr. Uklanjanje linije ako se radi o jednoj problemati?noj liniji)

NAPOMENA: Ne?e se pojavljivati sintaksne greške, greške u imenima standardnih metoda, klasa, interface-a iz Java API-ja i sli?no, koje bi izazvale neuspješno kompajliranje.

a.1)

```
import java.io.Serializable;

class A1 implements I2
{
  static String a1s;
  A1 a1;
  {
    a1s = "a1";
    System.out.println("A1-N1");
  }
  public static void main(String[] args)
  {
    System.out.println("Pocetak!");
    A1 a1 = new A1();
    A2 a2 = new A2(new A1(new A1(new A1("A2"))));
    A3 a3 = new A3(a2);
    Serializable a4 = new A4();
    I2 a5 = new A5();
    I1 a6 = new A6((A1) a5);

    System.out.println(a1 == a2);
    System.out.println(a2 == a3);
    System.out.println(a3.equals(a1));
    System.out.println(((A1) a4).equals(a2));
    System.out.println(((Serializable) a5).equals(a6));
    System.out.println(((A2) a3).a1s = ((A5) a6).a1s);
    System.out.println(((A1) a4).a1s = ((A3) a2).a1s);
    System.out.println(((A2) a3).a1s = ((A2) a6).a1s);
    System.out.println(((A4) a6).a1s = ((A5) a6).a1s);
    System.out.println(a1 = a2);
    System.out.println(a6 = (I1) a1);
    System.out.println(a6.equals(a5));
    System.out.println(((A3) a4).equals(a2));
    System.out.println(((Serializable) a6).equals(a1));

    System.out.println("kraj!");
  }
  A1(String x)
  {
    x.replace("A", "B");
    System.out.println("A(\"" + x + "\")");
  }
  A1(A1 a1) {
    this.a1 = a1;
    System.out.println("A1(A1)");
  }
```

```
  A1() {
    this(new A1("A1"));
    System.out.println("A1");
  }
  public void A1()
  {
    System.out.println("A1");
  }
  static
  {
    System.out.println("A1-S");
  }{
    System.out.println("A1-N2");
  }
}

class A2 extends A1
{
  {
    this.a1 = null;
    System.out.println("A2-N");
  }

  protected A2() {
    // A3 a3 = new A3(new A2());
    I2 a1 = new A1(new A1("A2-A1"));
    System.out.println("A2");
  }

  static {
    System.out.println("A2-S2");
  }

  A2(A1 a1) {
    super(new A1());
  }

  static {
    System.out.println("A2-S1");
  }
}

class A3 extends A2 implements Serializable
{
  A2 a2;

  public void ma1n(String[] args)
  {
    System.out.println("main");
  }
  private A3()
  {
    if (this.a1 instanceof A1)
    {
      this.a1 = new A2();
      this.a2 = new A3();
    }
    System.out.println("A3");
  }

  private void A3(){
    System.out.println("A3");
  }
```

```
  A3(A2 a2)
  {
    this();
    if (a2 instanceof I1)
    {
      this.a2 = new A6(new A6(a2));
    }
    this.a1 = a2;
  }

  {
    System.out.println("A3-N");
  }
    static {
    System.out.println("A3-S");
  }

}

class A4 extends A3
{
  static {
    System.out.println("A4-S");
  }
  A4()
  {
    super(new A2());
    System.out.println("A4");
  }
  {
    if (this.a1 instanceof I1)
      this.a1 = new A6();
    System.out.println("A4-N");
  }
}

class A5 extends A4
{
  public static int c;
  {
    System.out.println("A5-N");
  }
  public A5()
  {
    super();
    this.a2 = new A2();
    this.a1 = a2;
    System.out.println("A5");
  }
}

public class A6 extends A5 implements I1
{
  {
    System.out.println("A6-N");
  }

  static {
    System.out.println("A6-S");
  }

  public A6(int c)
  {
```

```
    this.c += c;
    System.out.println("A6(" + c + ")");
  }

  public A6()
  {
    this(0);
    this.a1 = this;
    System.out.println("A6");
  }

  public A6(A1 a1)
  {
    this(1);
    this.a1 = a1;
    System.out.println("A6(A1)");
  }
}

abstract interface I1
{
  void ma1n(String[] s);
}

interface I2{}
```

a1.2)

```
class A1 extends A2 implements B1
{
  private A1 a1;
  class A5 {
    {
    System.out.println("A5-N");
    }
    A5()
    {
      System.out.println("A5");
    }
    A5(A5 a5)
    {
      a1 = new A1();
    }
  }

  {
    System.out.println("A1-N");
  }
  A1()
  {
    System.out.println("A1");
  }

  A1(A1 a1)
  {
    this.a1 = a1;
  }

  public static void main(String[] args)
  {
    System.out.println("start");
    A4 a1 = new A1();
    A4 a2 = new A2(new A1());
    A4 a3 = new A3(new A2(new A1()));
```

```
    A4 a4 = new A4(new A3(new A2(new A1())));
    A1.A5 a5 = ((A1) a2).new A5();

    System.out.println(a2.a1 = a1.a1);
    System.out.println(a2.a1.equals(a2));
    System.out.println(a1.a1.equals(a2.a1));
  }

  static {
    System.out.println("A1-S");
  }
}

class A2 extends A4 implements B1
{
  static {
    System.out.println("A2-S");
  }
  {
    System.out.println("A2-N");
  }
  A2()
  {
    this(new A2("String constructor"));
    System.out.println("A2");
  }
  A2(String s)
  {
    System.out.println("A2(S)");
  }
  A2(A2 a2)
  {
    super();
    System.out.println("A2(A2)");
  }
}


class A3 extends A4 implements B1
{
  A3 a1 = new A3();
  A4 a4 = new A4();
  static {
    System.out.println("A3-S");
  }
  {
    System.out.println("A3-N");
  }
  A3()
  {
    System.out.println("A3");
  }
  A3(A4 a4)
  {
    this();
    System.out.println("A3(A4)");
    this.a1 = null;
  }
}

public class A4 implements B1
{
  A1 a1;
```

```
  A4 a2;
  static {
    System.out.println("A4-S");
  }
  {
    if (!(this instanceof A3))
    {
      if (0 == 1)
        a2 = new A4();
      System.out.println("A4-N");
    }
  }
  A4()
  {
    if (!(this instanceof A3))
      System.out.println("A4");
  }
  A4(A4 a4)
  {
    System.out.println("A4(A4)");
  }

  public boolean equals(A4 o) throws NullPointerException
  {
    if (super.equals(o))
      System.out.println("false");
    else
      System.out.println(true);

    return (super.equals((A4) o));
  }
}

interface B1
{
  boolean equals(Object o);
}
```

a.3)

```
import java.util.Arrays;
import java.io.Serializable;
class A1 extends A2
{
  private static final int Const = 10;

  public static void main(String[] argumenti)
  {
    A4[] a = new A4[Const];

    for(int i=0;i<a.length/3;i++)
    {
      if(Const%2<3)
        if(a.length>5)
        a[a.length-i-1] = new A4();
      a[i] = new A3();
      a[i+1] = new A2(a[i]);
      a[i+2] = new A1();
    }

    Arrays.stream(a).forEach(x ->
                             {
      x = new A4();
      x.a4 = new A4();
      if (x instanceof A3)
```

```
      {
        Serializable b = (Serializable) x;
        if (x.equals(b) == true)
          b = new A3();
      }
    });

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

class A2 extends A3
{
  void A2() {
    System.out.println("A2");
  }
  public static void main(String[] args)
  {

  }

  A2(A4 a4)
  {
    System.out.println("A2(A4)");
  }
}

class A3 extends A4 implements Serializable
{
  {
    System.out.println("A3-N");
  }
}

public class A4
{
  A4 a4;
  void A4()
  {
    System.out.println("A4");
  }
  static
  {
    System.out.println("A-S");
  }
}
```

b.1)

```
import java.util.*;

abstract class B1 extends B2 implements BI, BI2
{
  B1(){System.out.println("B1");};
  public void metoda()
  {
    System.out.println("B1-metoda");
  }

  public static void main(String[] args)
  {
```

```
    BI4 b1 = (bi2) -> {System.out.println("lambda1"); bi2.metoda();};
    BI b2 = new B2() {
      public void metoda2(BI2 bi2)
      {
        System.out.println("B2-metoda2");
      }
    };
    BI b3 = new B3();
    BI3 b4 = (BI3) (new B4());
    BI1 bi1 = new BI11();
    B5.B51 b5b51 = new B5.B51(){};
    BI5 bi5 = (bi2) -> {System.out.println("lambda2"); };
    BI21 bi21 = new BI21();

    bi21.metoda2(b4);
    BI2[] arr = {b1,(BI2) b2,(BI2) b3,b4, new B3()};
    // BI3[] arr2 = {(BI3) b1,(BI3)b2,(BI2)b3,(BI3)b4};
    bi5.metoda();
    bi1.metoda();

    for(BI2 x : arr)
    {
      x.metoda();
    }

    BI4[] arr2 = {b1, (BI4) b4, bi5};
    Arrays.stream(arr2).limit(5).forEach(x->x.metoda2(new B4()));
  }
}

abstract class B2 implements BI2, BI5
{
  abstract interface B2I
  {
  }
  B2(){System.out.println("B2");};

  public int metoda2(){return 1;}
}

class B3 implements BI3
{
  B3(){System.out.println("B3");};
}

class B4 extends B1 implements BI, BI2, BI3
{
  B4(){System.out.println("B4");};

  // static void metoda2(){System.out.println("B4-metoda2-S");}

  // public void metoda2(BI3 bi2) {
  public void metoda2(BI2 bi2) {
    System.out.println("B4-metoda2");
  }
}

abstract class B5 implements BI3
{
  // abstract static class B51 extends B5 implements BI1 {
  abstract static class B51 extends B5 implements BI3 {
    B51()
    {
```

```
      System.out.println("B51");
    }
  }
  {
    System.out.println("B5");
  }
  public void metoda()
  {
    System.out.println("B5-metoda");
  }
}


public interface BI extends BI1
{
  static class BI1 implements B2.B2I
  {
    void metoda()
    {
      System.out.println("BI1-metoda");
    }
  }

}

interface BI1 {
  default void metoda()
  {
    System.out.print("Test");
  }
  class BI11 extends BI.BI1{

  }
}

interface BI2 extends BI{
  class BI21 {
    private static int c;
    public void metoda2(BI2 bi2)
    {
      System.out.println("BI21-metoda2");
      c--;
      bi2.metoda();
      this.metoda();
    }
    private void metoda()
    {
      if (c != 0)
      {
        System.out.println("BI21-metoda");
        new BI211().metoda();
      }
      c--;c--;
    }
    class BI211 implements BI {
      public void metoda()
        // void metoda()
      {
        System.out.println("BI211-metoda");
        c++;
        BI21.this.metoda();
      }
      BI211()
```

```
      {
        System.out.println("BI211");
      }
      BI11 bi11 = new BI11();
    }
  }
}
interface BI3 extends BI, BI2 {
  default void metoda()
  {
    System.out.println("BI3-metoda");
  }
}
interface BI4 extends BI3 {
  default void metoda()
  {
    System.out.println("BI4-metoda");
  }

  void metoda2(BI2 bi2);
}

interface BI5 extends BI3, BI4
{
  // public void metoda();
}
```

c.1)

```
class C1 extends C2
{
  static int c;
  public static void main(String [] args)
  {
    try {
      C1 c1 = new C1();
      switch(c){
        case 0:
          try{
          C2 c2= new C2();
          c2.metoda();
        } catch (CE2 ex) {
          System.out.println("CE2");
        } catch (CE1 ex) {
          System.out.println("CE1");
        } catch (RuntimeException ex) {
          System.out.println("RE");
        } catch (Exception ex) {
          System.out.println("E");
        } catch (Throwable ex) {
          System.out.println("T");
        }
        case 1:
          c1.metoda();
          throw new Error();
        default:
          throw new CE2();
      }
    } catch (CE2 ex)
    {
      try{
        (new C3()).metoda();
      }
      catch(Exception exc) {
        System.out.println("fail");
```

```
      }
      try(C2 c22 = new C2()){}
      System.out.println("ex-CE2");
    } catch (CE1 ex)
    {
      System.out.println("ex-CE1");
    } catch (RuntimeException | Error ex)
    {
      System.out.println("ex-RE");
    } catch (Throwable ex)
    {
      System.out.println("ex-T");
    } finally {try{(new C1()).metoda();}catch(Exception ex){}};

  }

  public void metoda() throws Exception
  {
    System.out.println("C1 metoda");
    if(c>0){
      main(new String[5]);
      c = -65535;
    } else throw new CE1();
    throw new CE2();
  }


  C1()
  {
    c+=++c+c++;
    System.out.println("C1");
  }
}

class C2 extends C3
{
  public void metoda() throws Exception
  {
    System.out.println("C2-metoda");
    throw new CE1("Greska!");
  }
  public void close() throws CE1
  {
    System.out.println("C2-close");
  }
}

class C3 implements AutoCloseable, I1
{
  public void close() throws Exception
  {
    System.out.println("C3-close");
  }

  public void metoda() throws Exception
  {
    try
    {
      throw new CE2();
    }
    catch(Exception ex)
    {}
  }
```

```
}

//class CE1 extends Throwable {
class CE1 extends Exception {
  CE1()
  {
    this("CE1");
    System.out.println("CE1");
  }

  CE1(String str)
  {
    super(str);
    System.out.println("CE1(S)");
  }
}
class CE2 extends CE1 {
  CE2()
  {
    this("CE2");
    System.out.println("CE2");
  }

  CE2(String str)
  {
    super(str);
    System.out.println("CE2(S)");
  }
}

interface I1
{
  void metoda() throws Exception;
}
```

c.2)

```
class C1 extends C2
{
  static int c = 0;

  C1() throws CE2, CE3
  {
    System.out.println("C1");
    throw new CE1();
  }
  public static void main(String[] args) throws CE2, CE3
  {
    System.out.println("main");
    if (c++==0)
    {
      try {
        C1 c1 = new C1();
      } catch (Exception ex)
      {
        C2 c2 = new C2();
      }
    } else
    {
      throw new CE2();
    }
  }
}
```

```
class C2
{
  C2() throws CE3
  {
    System.out.println("C2");
    this.main();
  }

  public void main() throws CE1, CE3
  {
    System.out.println("C2-main");
    if (C1.c==0)
    {
      try {
        C1 c1 = new C1();
      } catch (Exception ex)
      {
        C2 c2 = new C2();
      }
    } else
    {
      throw new CE3();
    }

  }
}
class CE1 extends CE3
{
  CE1(String msg)
  {
    super(msg);
    System.out.println("CE1(S)");
  }

  CE1()
  {
    this("CE1");
    System.out.println("CE1");
  }
}

class CE2 extends Exception
{
  CE2(String msg)
  {
    super(msg);
    System.out.println("CE2(S)");
  }

  CE2() throws CE1
  {
    this("CE2");
    try {
      if (C1.c<0)
        throw new CE3();
      else throw new CE1();
    } finally {
      System.out.println("CE2");
      return;
    }
  }
}
```

```
class CE3 extends Exception
{
  CE3(String msg)
  {
    super(msg);
    System.out.println("CE3(S)");
  }

  CE3()
  {
    this("CE3");
    System.out.println("CE3");
  }
}
```

d.1)

```
public class B1 extends B2 implements BI1, BI2, BI3
{
  B1()
  {
    System.out.println("B1");
  }
  B1(String s)
  {
    this();
    System.out.println("B1("+s+")");
  }

  public void metoda()
  {
    System.out.println((new B2()).metoda("B1 metoda"));
  }

  public static void main(String args[])
  {
    BI1[][] b[]= new BI1[SYSTEM_CONST][][];
    for(int i = 0; i < b.length; i++)
    {
      b[i] = new BI1[b.length / 2][];
      for (int j = 0; j <= b.length /2; j++)
      {
        BI1[] br = b[i][j] = new BI1[b.length * SYSTEM_CONST/2];
        for (int k = 1; k >= 0 && k < br.length; k--)
        {
          br[k] = (BI3) (new B1());
          k+=2;
          if (k > 0 && k < br.length)
          {
            br[k-1] = new B2();
            ((B1) br[k-2]).metoda();
            // System.out.println(new B1(k));
            System.out.println(new B1(Integer.toString(k)).metoda("B1"));
          }
          else k-=2;
          br[k-=2].metoda();
        }
      }
    }
  }

  static protected String metoda(String x)
  {
```

```
    System.out.println("B1 metoda-S");
    return x.replace ("B1", "B");
  }

}

class B2 implements BI2
{
  private static int c;
  public void metoda()
  {
    System.out.println("B2-metoda");
  }
  static protected String metoda(String x)
  {
    c++;
    System.out.println("B2 metoda-S");
    x.replace("B1","B2");
    return x;
  }

  static void metoda2(String str) {
    System.out.println("B2-metoda2-S " + str);
  }
}

interface BI1 {
  static int SYSTEM_CONST = 5;
  public void metoda();
}
interface BI2 extends BI1{
  default void metoda(){
    System.out.println("BI2 metoda");
  };
}
interface BI3 extends BI1, BI2 {
  default void metoda2(String str){
    System.out.println("BI3-metoda2 " + str);
  }
  void metoda();
}
```

e.1)

```
public class E1 implements Runnable
{
  static volatile int c = 0;

  E1()
  {
    System.out.println("E1");
  }
  public static void main(String[] args)
  {
    System.out.println("Pocetak");
    new Thread(new E1()).run();
    new Thread(new E2()).run();
    System.out.println("parallel");
    metoda();
    new Thread(new E3()).run();
    new Thread(new E4()).run();
    System.out.println("parallel 2");
    metoda();
  }

  public void run()
  {
    if (c<3)
    {
      System.out.println("E2.run " + ++c);
      return;
    }
  }

  public static void metoda()
  {
    Thread[] t = {new Thread(new E1()), new Thread(new E2())};

    for(Thread thr : t)
    {
      thr.start();
    }

    Thread e3 = new Thread(new E3(t[1]));
    Thread e4 = new Thread(new E4(e3));
    e3.start();
    e4.start();
    try {
      t[1].join();
    }
    catch (InterruptedException ex)
    {}

    final I1 r2 = ()->{
      System.out.println("r2 run");
      for(int i = 0; i < 5; i++)
      {
        System.out.println(i);
      }

      try {
        e4.join();
      }
      catch (InterruptedException ex)
      {}
    };
```

```
    Thread x = new Thread(r2);
    final Runnable r1 = ()->{
      System.out.println("r1 run");
      for(int i = 5; i > 0; i--)
      {
        System.out.println(i);
      }

      try {
        x.join();
      }
      catch (InterruptedException ex)
      {}
    };
    Thread y = new Thread(r1);
    x.start();
    y.start();

    t[2] = new Thread(new E3());
    try {
      t[2].join();
    }
    catch (InterruptedException ex)
    {}
  }
}

class E2 extends E1
{
  public void run()
  {
    if (c<3)
    {
      System.out.println("E2.run " + ++c);
      return;
    }
  }
}

class E3 extends E4
{
  Thread e2;
  E3(){}
  E3(Thread e2)
  {
    this.e2 = e2;
    System.out.println("E3");
  }

  public void run()
  {
    System.out.println("E3.run");
    try{
      e2.join();
      System.out.println("E3-E2-join");
    }
    catch(InterruptedException ex)
    {}
    finally {
      System.out.println("E3 end");
    }
  }
```

```
}

class E4 extends Thread
{
  Thread e3;
  E4()
  {
    System.out.println("E4");
  }
  E4(Thread e3)
  {
    System.out.println("E4(E3)");
  }
  public void run()
  {
    System.out.println("E4 run");
    try{
      e3.join();
      System.out.println("E4-E3-join");
    }
    catch(InterruptedException ex)
    {}
    finally {
      System.out.println("E4 end");
    }
  }
}

interface I1 extends Runnable
{
  static final int c = 1;
}
```