Programski jezici II
Test 2

1. Analizirati kod u sljedećim primjerima i utvrditi da li se može kompajlirati i izvršiti. Ako kod nije moguće kompajlirati ili izvršiti, označiti „problematične" linije koda i navesti razloge. Ako se kod može kompajlirati i izvršiti, napisati izlaze. Po potrebi detaljno obrazložiti. **(7 x 5 bod)**

a)

```java
// A1.java

public class A1 {
    A1 a1;
    static {
        System.out.println("A1-S");
    }
    {
        System.out.println("A1-N");
    }

    public A1() {
        System.out.println("A1()");
    }
    public A1(A1 a1) {
        this.a1 = a1;
        System.out.println("A1(A1)");
    }
    protected void metoda1() {
        System.out.println("A1.metoda1()");
    }
    public void metoda2() {
        System.out.println("A1.metoda2()");
    }
}

class A2 extends A1 {
    static {
        System.out.println("A2-S");
    }
    private A1 a1;
    {
        a1 = new A1();
        System.out.println("A2-N");
    }
    protected A2() {
        super();
        System.out.println("A2()");
    }
    public A2(A1 a1) {
        super(a1);
        System.out.println("A2(A1)");
    }
```

```java
        @Override
        public final void metoda1() {
            System.out.println("A2.metoda1()");
        }
        public void metoda2() {
            System.out.println("A2.metoda2()");
        }
}

class A3 extends A2 {
    protected A2 a2;
    {
        a2 = new A2(new A1());
    }

    public A3() {
        System.out.println("A3()");
    }
    public void metoda2() {
        System.out.println("A3.metoda2()");
    }
}

class A4 extends A3 {
    public A4() {
        System.out.println("A4()");
    }

    public static void main(String[] args) {
        A4 a4 = new A4();
        a4.a2.metoda1();
        a4.metoda2();
    }
}
```

b)

```java
// B1.java

import java.util.*;

public class B1 {
    public static void main(String arr[]) {
        B2 b1 = new B2(-2);
        B2 b2 = new B2(-1);
        B2 b3 = new B2(0);
        B2 b4 = new B2(1);
        B2 b5 = new B2(2);

        int i = 1;
        for (B2 b : Arrays.asList(b1, b2, b3, b4, b5)) {
            System.out.println(b.reverse(i + ". " + b.toString()));
            System.out.println(b.concat(BI1.x, BI3.x));
            i++;
        }
    }
}

class B2 implements BI1 {
    static int count;
    {
        count++;
    }
    int id;

    public B2(int id) {
        if (id > 0 || new Random().nextDouble() > 1)
            this.id = id;
        else
            id = count;
        System.out.println(this + ": created");
    }
    public String toString() {
        return "B2(id = " + id + ")";
    }
    public String reverse(String str) {
        return str;
    }
}

interface BI1 extends BI2, BI3 {
    String x = "BI1";
    default String concat(String str1, String str2) {
        return str2 + str1;
    }
    String reverse(String str);
}
```

```java
interface BI2 {
    String x = "BI2";

    public default String concat(String str1, String str2) {
        return str1 + str2;
    }
}

interface BI3 {
    String x = "BI3";

    default String reverse(String str) {
        return new StringBuilder(str).reverse().toString();
    }
}
```

c)

```java
// C1.java

public class C1 {

    C1() {
        System.out.println("C1()");
    }

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

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

```java
class C2 {
    C2 () {
        System.out.println("C2()");
    }
    void metoda() throws CE1 {
        C3 c3 = new C3();
        System.out.println("C2: metoda()");
        c3.metoda();
    }
}

class C3 {
    C3 () {
        System.out.println("C3()");
    }
    protected void metoda() throws CE1 {
        System.out.println("C3: metoda()");
        throw new CE2("CE2");
    }
}

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);
    }
}
```

d)

```java
// D1.java

public class D1 implements DI3 {
    public D1() {
        System.out.println("D1()");
    }
    public static void main(String… args) {
        DI1 d1 = new D1();
        DI2 d2 = new D1();
        DI3 d3 = new D2();
        D1 d4 = new D2();
        D2 d5 = new D2();
        d1.metoda1();
        d2.metoda2();
        d3.metoda1();
        d4.metoda2();
        d5.metoda1();
    }
    void metoda2() {
        System.out.println("D1.metoda2()");
    }
}

class D2 extends D1 implements DI2 {
    public D2() {
        System.out.println("D2()");
    }
    public void metoda1() {
        System.out.println("D2.metoda1()");
    }
    public void metoda2() {
        System.out.println("D2.metoda2()");
    }
}

interface DI1 {
    public default void metoda1() {
        System.out.println("DI1.metoda1()");
    }
}

interface DI2 extends DI1 {
    void metoda1();
    default void metoda2() {
        System.out.println("DI2.metoda2()");
    }
}

abstract interface DI3 extends DI2 {
    final int c = 10;
    default void metoda1() {
        System.out.println("DI3.metoda1()");
    }
    abstract void metoda2();
}
```

e)

```java
// E1.java

public class E1 extends Thread implements EI {
    static int c = 1;
    {
        c++;
    }
    int id;

    public E1() {
        id = c;
        if (id % 2 == 0)
            setDaemon(true);
        else
            setDaemon(false);
    }

    public static void main(String argv[]) throws Exception {
        Runnable[] niz = {new E1(), new E1(),
            new Thread(), new E1(), new Thread()};
        System.out.println("First line");
        for (Runnable r : niz) {
            System.out.println("Checking...");
            if (r instanceof Thread) {
                Thread t = (Thread) r;
                if (t.isDaemon()) {
                    System.out.println("Starting background thread...");
                    new Thread(r);
                } else {
                    if (t instanceof EI)
                        ((E1) t).run("arg1");
                    else
                        t.run();
                }
            } else {
                new Thread(r).start();
            }
        }
        System.out.println("Last line");
    }

    public void run() {
        for (int i = 1; i < 6; i++)
            System.out.println("E1(" + id + "): " + i);
    }
}

interface EI extends Runnable {
    default void run(String... args) {
        if (args.length > 0) {
            System.out.println(args[0]);
            new Thread(this).start();
        }
    }
}
```

f)

```java
// F1.java

import java.util.Arrays;

public class F1<T extends Runnable>
    extends F2<Runnable> implements FI {;
    public static void main(String args[]) {
        FI f1 = new F1<Thread>();
        F1<F1> f2 = new F1<>();
        Runnable f3 = new F1<F1>();
        F2<Runnable> container = new F2<>(f1, f2, f3);

        System.out.println(container.toString());
        container.runAll();
        System.out.println("End");
    }
    public String toString() { return "T1"; };
}

class F2<T extends Runnable> implements FI {

    T[] arr;

    public F2(T... elements) {
        arr = elements;
    }
    public void runAll() {
        for (T e : arr) {
            new Thread(e).start();
        }
    }
    public String toString() {
        return Arrays.asList(arr).toString();
    }
}

interface FI extends Runnable {
    default void run() {
        System.out.println("Running...");
    }
}
```

g)

```java
// G1.java

import java.io.*;

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 + "\n" + g22.b);
        G3 g33 = (G3) cin.readObject();
        System.out.println(g33.a + "\n" + g33.b);
        cin.close();
    }
}
class G2 implements Externalizable {
    private 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();
    }
}
```

2. Prikazati stanje memorije u trenutku izvršavanja linije sa oznakom 1. Pretpostaviti da je rezervisana dovoljna veličina heap-a i da se garbage collector odmah izvršava nakon poziva metode gc(). **(8 bod)**

```java
// Memory.java

public class Memory {
    int id;
    double[] doubles = new double[10000000];
    long[] longs = new long[6000000];
    Memory[] mys = new Memory[3];
    Memory m;
    Dumb d;

    public Memory(Memory m) {
        this.m = m;
        if (m != null)
            id = m.id + 1;
    }

    public static void main(String[] args) {
        Memory m0 = new Memory(null);
        Memory m1 = new Memory(m0);
        Memory m2 = new Memory(m1);
        Memory m3 = new Memory(m2);
        Memory m4 = m3;
        Memory m5 = new Memory(m0);
        Memory[] ms = new Memory[3];
        ms[0] = new Memory(m2);
        ms[1] = new Memory(m3);
        ms[2] = new Memory(m2);
        m1 = m2 = m3 = m5 = null;
        System.gc();
        ms[0] = new Memory(ms[0]);
        ms[1] = new Memory(ms[1]);
        ms[2] = new Memory(ms[2]);
        ms[2].m = ms[0];
        m3 = new Memory(ms[0] = null);
        ms[0] = new Memory(ms[0]);
        System.gc(); // 1
    }
}

class Dumb {
    float[] floats = new float[20000000];
    int[] ints = new int[40000000];
}
```

3. Analizirati kod u sljedećim primjerima i utvrditi da li se može kompajlirati i izvršiti. Ako kod nije moguće kompajlirati ili izvršiti, navesti razloge. Ako se kod može kompajlirati i izvršiti, napisati izlaze. Zadaci se boduju po principu "sve ili ništa". **(7 x 1 bod)**

a)

```java
public class Klasa1 {
    public static void main(String []args) {
        int i = 0;
        int j = 10;
        int k = 0;
        while (i++ < 5 || --j > 3) {
            k += i + j;
        }
        System.out.println(k);
    }
}
```

b)

```java
;public class Klasa2 {;
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++)
            new Thread(() -> System.out.println("Thread" + i)).start();
    };;
}
```

c)

```java
public class Klasa3 {
    public static void main(String x[]) {
        Object obj = new String("Word");
        String str = "Word";
        StringBuilder sb = new StringBuilder("Word");

        System.out.println(obj.equals(str));
        System.out.println(obj == "Word");
        System.out.println(str == "Word");
        System.out.println(str == sb.toString());
    }
}
```

d)

```java
import java.util.stream.*;

public class Klasa4 {
    public static void main(String[] args) {
        Stream.iterate(1, e -> e + 1)
                .filter(Klasa4::isPrime)
                .limit(5)
                .forEach(System.out::println);
    }
    public static boolean isPrime(int number) {
        return number > 1 &&
            IntStream.range(2, number)
                    .noneMatch(i -> number % i == 0);
    }
}
```

e)

```java
public class Klasa5 {

    public static main(String[] args) {
        for (String s : args)
            System.out.println(s);
    }

    public static main(String arg) {
        System.out.println(arg);
    }
}
```

f)

```java
import java.util.NoSuchElementException;

public class Klasa6 {

    int[] niz;

    Klasa6(int... elements) {
        niz = elements;
    }

    public static void main(String[] args) {
        Klasa6 k6 = new Klasa6(1, 2, 3, 5, 6, 7, 8);
        Klasa6.Iterator itr = k6.iterator();
        while (true)
            System.out.println(itr.next());
    }

    public Iterator iterator() {
        return new Iterator();
    }

    class Iterator {
        int pos;
        int next() {
            try {
                pos++;
                return niz[pos];
            } catch (ArrayIndexOutOfBoundsException e) {
                pos--;
                throw new NoSuchElementException("Kraj niza");
            }
        }
    }
}
```

g)

```java
public class Klasa7 {
    public static void main(String[] args) {
        return;
        System.out.println("main");
    }
}
```