Elektrotehnički fakultet
Banja Luka
Katedra za računarstvo i informatiku

Programski jezici II
Test 1

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)

```
// A1.java

import java.io.Serializable;

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

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

    private A1 a1;

    public A1() {
        System.out.println("A1()");
    }
    public A1(A1 a1) {
        this();
        System.out.println("A1(A1)");
        this.a1 = a1;
        new A2(a1);
    }
    public void metoda1() {
        System.out.println("A1.metoda1()");
    }
}

class A2 extends A1 implements Serializable {
    protected A2() {
        System.out.println("A2()");
    }

    public A2(A1 a1) {
        System.out.println("A2(A1)");
        a1.metoda1();
    }
    @Override
    public void metoda1() {
        System.out.println("A2.metoda1()");
    }
    public void metoda2() {
        System.out.println("A2.metoda2()");
    }
}
```

```
class A3 extends A2 {

    A2 a2 = null;

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

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

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

    public A3(A2 a2) {
        this.a2 = a2;
        System.out.println("A3(A2)");
    }

    public A3(A1 a1, A2 a2) {
        this(a2);
        System.out.println("A3(A1,A2)");
    }

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

class A4 extends A3 {

    A1 a1 = new A1();
    A3 a2 = new A3(new A1(new A1()), new A2(a1));
    Serializable a3 = new A3();

    public A4() {
        super();
        System.out.println("A4()");
        super.metoda1();
    }

    public static void main(String[] args) {
        A4 a4 = new A4();
        a4.metoda1();
        a4.metoda2();
        ((A2) a4).metoda1();
        ((A2) a4.a3).metoda2();
    }
}
```

b)

```
// B1.java

public class B1 implements BI2 {

    String j = "j";

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

    public void print(String str) {
        System.out.println(str.isEmpty() ? "" : str.charAt(0));
    }

    public static void main(String... argv) {
        B1 b1 = new B1();
        B1 b2 = new B2();
        BI1 b3 = new B3("b3");
        BI2 b4 = new B2();

        b1.print(b1.j);
        b2.print("av");
        b2.close();
        b4.print("b3");
        ((BI1) b3).close();
        try (B3 b5 = new B3("a")) {
            b5.print("Zadnja linija?");
        }
    }
}

class B2 extends B1 {

    static BI1 b1 = new B1();
    String str = "";

    public B2() {
        System.out.println("B2()");
    }

    public void close() {
        System.out.println("B2 closed...");
    }
    public void print(String str) {
        str = str + 'a';
        System.out.println(str);
    }
}
```

```
interface BI1 extends AutoCloseable {

    default void close() {
        System.out.println("Resource closed...");
    }

    class B3 extends B2 implements BI2 {

        String str;

        B3(String x) {
            str = x;
        }

        public void print(String x) {
            StringBuilder builder = new StringBuilder(x + str);
            System.out.println(builder.reverse());
        }
    }
}

abstract interface BI2 extends BI1 {
    public void print(String str);
}
```

c)

```
// 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 (CE1 e) {
            System.out.println("main 2: " + e);
        } catch (CE2 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");
        }
    }
}
```

```
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 Error {
    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)

```
// D1.java

public class D1 implements Comparable<D1> {

    static D1 instance;
    String rijec;

    private D1() {}

    public static void main(String arr[]) {
        D1 d1 = D1.getInstance();
        D1 d2 = D1.getInstance();

        d1.rijec = "hello";
        d2.rijec = new String("HELLO");
        System.out.println(d1.rijec == d2.rijec);
        System.out.println(d1.rijec.equalsIgnoreCase(d2.rijec));
        System.out.println(d1.compareTo(d2));
        System.out.println(d1.rijec + " ? " + d2.rijec);
    }


    public static D1 getInstance() {
        if (instance == null)
            instance = new D1();
        return instance;
    }

    @Override
    public int compareTo(D1 other) {
        other.rijec.toLowerCase();
        return rijec.compareTo(other.rijec);
    }

}
```

e)

```
// E1.java

public class E1 extends Thread {

    int id = 0;

    public E1() {
        super();
    }

    public static void main(String[] args) {
        System.out.println("MainThread");
        new E1().start();
    }

    public void run() {
        System.out.println("Start...");
        E1[] niz = {new E2(1), new E2(2), new E3("3"),
            new E2(4), new E3("5")};
        for (E1 e : niz) {
            if (e.isDaemon()) {
                new Thread(e).start();
            } else {
                System.out.println("New thread...");
                e.run();
                e.start();
            }
        }
        System.out.println("End");
    }

}

class E2 extends E1 implements Runnable {

    public E2(int id) {
        this.id = id > 1 ? id : this.id;
        if (this.id == id) {
            setDaemon(true);
        }
    }

    @Override
    public void run() {
        System.out.println("E2 - " + id + ": " + isDaemon());
        for (int i = 1; i < 6; i++) {
            System.out.println("E2 - " + id + ": " + i);
        }
    }

}

class E3 extends E2 {

    public E3(String s) {
        super(new Integer(s + ""));
    }
}
```

f)

```
// F1.java

public class F1<T1, T2> implements FI<T2> {
    T2 t2;

    public static void main(String args[]) {
        F1<String, Integer> f1 = new F1<>();
        FI<Double> f2 = new F1<>();
        F2<Integer, F1> f3 = new F2<>();
        F3<Object, String> f4 = new F3<>(1.5, f3);

        f1.print(f4.x);
        f2.print((double) f1.t2);
        f1.t2 = 3;
        f3.s = "" + f1.t2;
        System.out.println(f3.getModified(2));
        System.out.println(f4.getSuperModified("" + f1.t2));
    }

    @Override
    public void print(T2 t) {
        System.out.println(t);
    }
}

class F2<T1 extends Number, T2> {
    String s;

    public F2() {
        s = "a";
    }

    public char getModified(int i) {
        return (char) (s.charAt(0) + i);
    }
}

class F3<T, T2> extends F2<Integer, F3> {
    int x;

    public F3(double d, F2 f) {
        x += f.getModified((int) d);
    }

    public String getSuperModified(String num) {
        return "" + new Double(new Integer(x + num) * 0.1);
    }
}

interface FI<T> {
    void print(T t);
}
```

g)

```
// G1.java

import java.io.*;

public class G1 {

    public static void main(String[] args) throws Exception {
        G3 g1 = new G3();
        G2 g2 = new G2(g1);

        try (ObjectOutputStream o = new ObjectOutputStream(
                 new FileOutputStream("data.bin"))) {
            o.writeObject(g1);
            o.writeObject(g2);
        } finally {
            System.out.println("Data saved");
        }

        try (ObjectInputStream in = new ObjectInputStream(
                 new FileInputStream("data.bin"))) {
            g1 = (G3) in.readObject();
            g2 = (G2) in.readObject();
        } finally {
            System.out.println("Data loaded");
        }

        System.out.println(g2.g3.i);
    }
}

class G2 implements Serializable {

    G3 g3;

    public G2() {
        System.out.println("G2()");
    }

    public G2(G3 g3) {
        if (g3 != null)
            this.g3 = g3;
        System.out.println("G2(G3)");
    }

    public void writeExternal(ObjectOutput out)
        throws IOException {
        System.out.println("G2.writeExternal");
    }

    public void readExternal(ObjectInput in)
        throws IOException, ClassNotFoundException {
        System.out.println("G2.readExternal");
        g3 = new G3();
    }
}
```

```
class G3 extends G2 implements Externalizable {

    int i = 0xaaa;

    public G3() {
        super();
        System.out.println("G3()");
    }

    public void writeExternal(ObjectOutput out)
        throws IOException {
        super.writeExternal(out);
        System.out.println("G3.writeExternal");
    }

    public void readExternal(ObjectInput in)
        throws IOException, ClassNotFoundException {
        super.readExternal(in);
        System.out.println("G3.readExternal");
    }
}
```

2. Napisati izlaz sljedećeg programa i 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)**

```
// M1.java

import java.util.*;

public class M1 {
    int id;
    M1 m1;
    M2 m2;
    char[] nizChar = new char[10_000_000];
    int[] nizInt = new int[5_000_000];
    M2 mArray[][] = new M2[3][2];

    public M1(int id) {
        System.out.println("M1: " + id);
        this.id = id;
    }

    public M1(M1 m1,int id,M2 m2) {
        System.out.println("M1: " + id);
        this.m1 = m1;
        this.m2 = m2;
        this.id = id;
    }

    @Override
    protected void finalize() {
        System.out.println(id + "finallize");
    }

    public static void main (String[] args) {
        M1 m10 = new M1(10);
        M2 m21 = new M2(m10,21);
        M1 m11 = new M1(m10,11,m21);
        M2 m22 = new M2(null,22);
        M1 m12 = new M1(null,12,m22);
        m12.mArray[0][0] = m22;
        m12.mArray[1][1] = new M2(23);
        m12.mArray[2][1] = new M2(24);
        m12.mArray[1] = null;
        System.gc();
        m10.mArray[1][0] = new M2(25);
        m11.m2 = m10.mArray[1][0];
        m11 = null;
        System.gc();
        m10.mArray[2][2] = new M2(new M1(1000),26);
        System.gc(); // 1
    }
}
```

```
class M2 {
    float[] f = new float[2_500_000];
    M1 m1 = new M1(0);
    private int id2 = 0;

    public M2(M1 m1,int id2) {
        System.out.println("M2: " + id2);
        this.m1 = m1;
        this.id2 = id2;
    }

    public M2(int id2) {
        System.out.println("M2: " + id2);
        this.id2 = id2;
    }

    @Override
    protected void finalize() {
        System.out.println(id2 + "finallize");
    }
}
```

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)

```
public class Test1 {
    public static void main(String... a) {
        Klasa1 k1 = new Klasa1() {
            public void add(int a, int b) {
                return a + b;
            }
            public void sub(int a, int b) {
                return a - b;
            }
        };
        System.out.println(k1.add(10,20));
        System.out.println(k1.add(5,10));
    };;;
    ;

    static abstract class Klasa1 {
        public abstract add(int x, int y);
        public abstract sub(int x, int y);
    }
}
```

b)

```
public class Klasa2 {

    private int x;

    public static void main(String[] args) {
        Klasa21 k2 = new Klasa21() {
            @Override
            public void print(String str) {
                System.out.println("Poruka");;;
            }
        };
        k2.print(x);
    }

}

class Klasa21 extends Klasa2 {
    public void print(String str) {
        System.out.println(str);
    }
}
```

c)

```
public class Klasa3 {
    public static void main(String args[]) {
        Integer i1 = 10;
        Integer i2 = new Integer(10);
        System.out.println(i1 == i2);
    }
}
```

d)

```
public class Klasa4 extends Thread {
    static int c = 0;

    public static void main(String... args) {
        if ((c++) == 0)
            main("arg1", "arg2", "arg3");
        else if (c == 1)
            main(new Integer(args[0].charAt(3)));
        else if (c == 2)
            main('A');
        else
            main();
    }
    public static void main() {
        System.out.println("Pocetak programa");
        new Thread() {
            public void run() {
                for (int i = 0; i < 5; i++);
                {
                    System.out.println("Nit");
                }
            }
        }.start();
    }
    public static void main(int i) {
        System.out.println(i);
        c++;
    }
    public static void main(char c) {
        char x = (""+c).toLowerCase().charAt(0);
        System.out.println(x);
        c++;
    }
}
```

e)

```
public enum Dan {
    PON(1), UTO(2), SRE(3), CET(4), PET(5), SUB(6), NED(7);

    int rb;
    Dan(int rb) {
        this.rb = rb;
    }

    public static void main(String[] args) {
        for (Dan e : Dan.values())
            System.out.println(e.rb + ". " + e);
    }
}
```

f)

```
public class Klasa6 {
    public static void main(String x[]) {
        int[][][] niz = {{{1, 2, 3}, {4, 5, 6}}};
        for (int i = 0; i < niz.length; i++)
            for (int j = 0; j < niz[1].length; j++)
                for (int k = 0; k < j; i++)
                    System.out.println((Math.random() > 0.5)
                                        ? niz[i][j][k] : "x");
    }
}
```

g)

```
public class Main {
    public static void main(String argv[]) {
        Klasa7 k7 = new Klasa7();
        System.out.println(k7.broj);
    }
}

class Klasa7 {
    private int broj = 1;

    Klasa7() {
        broj += broj;
    }
}
```