Elektrotehnički fakultet
Banja Luka
Katedra za računarstvo i informatiku

Programski jezici II
Test 3

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

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

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

    public A2() {
        super(new A1());
        System.out.println("A2()");
    }
    protected A2(A1 a1) {
        super();
        A1.a1 = a1;
        System.out.println("A2(A1)");
    }

    public void metoda1() {
        super.metoda1();
        System.out.println("A2: metoda1()");
```

```
    }
    public void metoda2() {
        System.out.println("A2: metoda2()");
    }
    protected void metoda3() {
        metoda1();
        System.out.println("A3: metoda3()");
    }
}

class A3 extends A2 implements Serializable {
    static {
        System.out.println("A3-S");
    }
    {
        System.out.println("A3-N");
    }

    public A3() {
        super();
        System.out.println("A3()");
    }
    public A3(A2 a2) {
        super(a2.a1);
        System.out.println("A3(A2)");
    }
    public void metoda3() {
        System.out.println("A3: metoda3()");
    }
}

class A4 extends A3 {
    A3 a3 = new A3(new A2(new A1()));

    public A4() {
        System.out.println("A4()");
    }
    public static void main(String[] args) {
        A4 a4 = new A4();
        a4.metoda1();
        a4.metoda2();
        a4.metoda3();
        a4.a3.metoda1();
        a4.a3.metoda2();
        a4.a3.metoda3();
    }
}
```

b)

```
// B1.java

public class B1 {
    B2 b2;
    private int x = 10;
    private int y = 5;

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

    static class B2 {
        B3 b3;
        int z = 5;

        public B2() { System.out.println("B2()"); }
        public int metoda() {
            return y + z;
        }
    }

    protected static class B3 {
        public B3() { System.out.println("B3()"); }

        public int metoda() {
            return x * 2;
        }

        class B4 extends B3 {
            public B4() { System.out.println("B4()"); }
            public void metoda(int n) {
                System.out.println(n);
            }
        }
    }

    protected int metoda() {
        return x;
    }

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

        System.out.println(
            b1.metoda() + "\n" +
            b1.b2.metoda() + "\n" +
            b3.metoda() + "\n" +
            b1.b2.b3.metoda());
    }
}
```

c)

```
// C1.java

public class C1 {

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

    public static void main(String[] args) {
        C1 c1 = new C1();
        try {
            System.out.println("main 1");
            c1.metoda();
        } catch (CE2 e) {
            System.out.println("main 2: " + e);
        } catch (CE1 e) {
            System.out.println("main 3: " + e);
        } catch (Exception e) {
            System.out.println("main 4: " + e);
        } catch (Throwable e) {
            System.out.println("main 5: " + e);
        }
    }

    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 {
        try {
            System.out.println("C3: metoda()");
            throw new CE2("CE2");
        } finally {
            System.out.println("finally");
        }
    }
}

class CE1 extends Exception {
    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)

```
// DI1.java

public interface DI1 {
    public static void main(String[] args) {
        new DI2() {
            public void metoda2() {
                System.out.println("Main: metoda2()");
            }
            void metoda3() {
                System.out.println("Main: metoda3()");
            }
        }.metoda1();
    }
}

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

interface DI3 extends DI2 {
    abstract void metoda1();
    default void metoda2() throws RuntimeException {
        System.out.println("DI3: metoda3()");
    }
    void metoda3();
}

interface DI4 extends DI2, DI3 {
    default void metoda1() {
        System.out.println("DI4: metoda1()");
    }
    void metoda2();
}
```

e)

```
// E1.java

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

public class E1 extends Thread {
    static List<E1> threads = new ArrayList<>();
    final static Random prng = new Random();
    static int c;
    int id;

    public E1() {
        id = (prng.nextDouble() > 1.0) ? c++ : 0;
        this.setDaemon(true);
        threads.add(this);
        new Thread(this).run();
    }

    public static void main(String[] args) {
        Thread[] niz = {new E1(), new E1(), new Thread(new E1())};
        System.out.println("Main start");
        int count = 0;
        for (int i = 0; i < niz.length; i++) {
            if (niz[i] instanceof E1) {
                System.out.println("Starting thread...");
                niz[i].start();
                count++;
            }
            if (count > 1) {
                System.out.println("Starting all threads...");
                runAll();
                break;
            }
        }
        System.out.println("Main end");
    }

    public void run() {
        Stream.iterate(1, e -> e + 1)
              .limit(5)
              .forEach(e -> System.out.println("E1(" + id + "): " + e));

    }

    public static void runAll() {
        threads.stream()
               .forEach(e -> e.start());
    }
}
```

f)

```
// F1.java

import java.util.*;

public class F1<T1, T2 extends Number, T3>
    extends F2<T1, T2> implements FI<T1> {

    T3 t3;

    public F1(T1 t1, T2 t2, T3 t3) {
        super(t1, t2);
        this.t3 = t3;
    }
    public String print() {
        return String.format("%s - %s - %s", t1, number, t3);
    }
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(1, 2, 3);
        FI<String> f1 = new F1<>("Java", 10, numbers);
        F2 f2 = new F2<List<Integer>, Double>(numbers, 1.5);
        F3 f3 = new F3<>(numbers.get(2));
        System.out.println(f1.print());
        System.out.println(f2.toPercentage());
        System.out.println(f3.getNumber());
    }
}

class F2<T1, T2 extends Number> {
    T1 t1;
    Number number;

    public F2(T1 t1, T2 number) {
        this.t1 = t1;
        this.number = number;
    }
    public String toPercentage() {
        return (number.doubleValue() * 100) + "%";
    }
}

class F3<T2 extends Number> extends F2<String, T2> {
    private T2 number;

    public F3(T2 number) {
        super(number.toString(), number);
        this.number = number;
    }
    public Number getNumber() {
        return number;
    }
}

interface FI<T1> {
    String print();
}
```

g)

```
// 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);
        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;
    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;
    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. Napisati izlaz sljedećeg programa i prikazati stanje memorije u trenutku izvršavanja linija sa oznakama 1 i 2. Pretpostaviti da je rezervisana dovoljna veličina heap-a i da se *garbage collector* odmah izvršava nakon poziva metode *gc()*. **(8 bod)**

```
// Memory.java

public class Memory {
    int id;
    char[] chars = new char[10_000_000];
    long[] longs = new long[6_000_000];
    Memory[] mys = new Memory[3];
    Dumb d;
    Memory m;

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

    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(); // 1
        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(); // 2
    }
}

class Dumb {
    float[] floats = new float[20_000_000];
    int[] ints = new int[40_000_000];
}
```

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)

```
import java.util.*;

public class Klasa1 {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        System.out.println(
            numbers.stream()
                   .filter(e -> e % 2 == 0)
                   .findFirst()
                   .orElse(0));
    }
}
```

b)

```
public class Klasa2 {
    public static void main() {
        new Thread(() -> print("Hello")).run();
        System.out.println("World!");
    }

    static void print(String str) {
        System.out.print(str);
    }
}
```

c)

```
import java.util.function.*;

public class Klasa3 {
    public static void main(String[] args) {
        int x = 10;
        Supplier<Integer> temp = () -> compute(x);

        if (x > 15 && temp.get() < 50) {
            System.out.println("Correct");
        } else {
            System.out.println("Incorrect");
        }
    }

    static int compute(int number) {
        System.out.println("Computing...");
        return 2*number + 1;
    }
}
```

da je obrnuto, ispis bi bio:
Computing...
Incorrect

inače je:
Incorrect

d)

```
import java.util.stream.Stream;

public class Klasa4 {
    public static void main(String[] args) {
        Stream.iterate(1, e -> e * 2)
              .filter(Klasa4::isGT10)
              .limit(50);
    }

    public static boolean isGT10(int number) {
        System.out.println("isGT10 " + number);
        return number > 10;
    }
}
```

ništa se ne ispisuje

e)

```
import java.util.*;

public class Klasa5 {
    public static void main(String a[]) {
        List<Integer> list =
            Arrays.asList(1, 6, 3, 4, 8, 6, 10, 8, 10, 10);
        Set<Integer> set = new HashSet<>(list);
        System.out.println(list);
        System.out.println(set);
    }
}
```

[1, 6, 3, 4, 8, 6, 10, 8, 10, 10]

[1, 3, 4, 6, 8, 10]

f)

```
public class Klasa6 {
    private int x = 10;

    public static void main(String[] args) {
       int y = 15;
       while (x > 2 && (y--) > 5) {
           x--;
           System.out.println(x + " " + y);
       }
    }
}
```

kompajlerska greška, x nije static

g)

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

public class Klasa7 {
    public static List<Person> createPeople() {
        return Arrays.asList(
            new Person("Bob", 37),
            new Person("Alice", 25),
            new Person("Jake", 25),
            new Person("Ryan", 37),
            new Person("Jill", 24)
        );
    }
```

```
    public static void main(String[] args) {
        List<Person> people = createPeople();

        // format: {key1=[values...], key2=[values...], ...}
        System.out.println(
            people.stream()
                  .collect(groupingBy(Person::getAge)));
    }
}

class Person {
    final String name;
    final int age;

    public Person(String theName, int theAge) {
        name = theName;
        age = theAge;
    }
    public String getName() { return name; }
    public int getAge() { return age; }
    public String toString() {
        return String.format("%s -- %d", name, age);
    }
}
```