Elektrotehnički fakultet
Banja Luka
Katedra za računarstvo i informatiku

Programski jezici 2 – II dio
Termin: 08.09.2010. godine

Napomena: Vrijeme trajanja ispita je 60 minuta.

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.

   a)

```
// A1.java

package a1;
import a1.a3.A2;
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();
    }
}

// A2.java

package a1.a3;
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++);
    }
}
```

A3()
A2()
A3()
A2()
StackOverFlow

b)

```
// B1.java

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

    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...");
    }
}
```

B1()
B2()
B3()
B3 metoda...
B3 metoda...
B1 metoda...

jer je metoda u klasi B1 private i ne može se overrajdovati

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 (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");
        }
    }
}
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);
    }
}
```

C1()
C2()
C3()
C2: metoda()
C3: metoda()
CE1: CE2
CE2: CE2
finally
main 2: CE2: CE2

ispis CE2: CE2 je takav zato što se poziva implicitno njegova toString() metoda

za izuzetke podrazumijevano vraća: ime klase + ":" + poruka

d)

```
// D1.java

public class D1 extends D3 implements DI {
    public static void main(String[] args) {
        D3 niz[] = {new D3(), new D2(), new DI()};
        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 D2 metoda() {
        System.out.println("D3: metoda()");
        return new D3();
    }
}
interface DI {
    D3 metoda();
}
```

ne može se konvertovati iz DI u D3

ne može se kastovati iz D2 u D1

ne može D2 da referencira D3, ali može obrnuto, tj. da roditelj referencira dijete

e)

```
// F1.java
public class F1 {
    public static void main(String[] args) {
        FT1<F2> ft1 = new FT1<F2>();
        ft1.metoda(new F2("Java"));
        System.out.println(ft1.t.s);
    }
}
class FT1<T> implements FTI<G> {
    T t;
    public void metoda(T t) {
        this.t = t;
    }
    public T metoda() {
        return t;
    }
}
class F2 {
    String s;
    F2(String s){
        this.s = s;
    }
}
interface FTI<G> {
    public void metoda(G t);
    public G metoda();
}
```

kompajlerska greška, mora biti T

f)

```
// E1.java

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 = 1; 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 = 1; i < 6; i++)
            System.out.println("E3 run")
    }

}
```

u ovom slučaju nema značaj

g)

```
// G1.java

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

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

    protected void metoda() throws Exception {
        test();
    }

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

abstract class G2 extends G1{
    abstract protected void metoda();
}

final class G3 extends G2{
    public void metoda() throws Exception{
        i += 10.51f;
        test();
    }
}
```

kompajlerska greška jer ne smije se u dijete klasi overrajdovati metoda tako da baca izuzetak ako u roditeljskoj ne baca, kao što ne može da baca "širi" tip izuzetaka

2. Napisati izlaz (2)

```
public class Test {
    public static void main(String[] args) {
        System.out.println(1 + 2 + "3");
        System.out.println("1" + 2 + 3);
    }
}
```

33
123

3. Odgovoriti:

a) Da li je *public main(int number) { }* validna metoda? (1)

b) Da li je *public static final main(int number) { }* validna metoda? (1)

c) Nestatičke promjenljive koje neće biti upisane u tok pri procesu serijalizacije deklarišu se kao… (1)

d) Da li se konstruktor može sinhronizovati? (1)

e) Šta je *reentrant* sinhronizacija? (1)

f) Nabrojati top-level Swing kontejnere. (1)