00.­­

// ThreadA.java

**publicclass** ThreadA **extends** Thread {

String name;

**public** ThreadA(String name) {

**this**.name = name;

}

**publicvoid** run() {

**new** Thread() {

**publicvoid** run() {

**for** (**int** i = 0; i < 10; i++) {

System.**out**.println(name + "2: " + i);

}

}

}.start();

System.**out**.println(name);

}

**publicstaticvoid** main(String args[])

**throws** InterruptedException {

System.**out**.println("Pocetak programa");

ThreadA a = **new** ThreadA("A");

ThreadB b = **new** ThreadB("B");

ThreadC c = **new** ThreadC("C", a);

a.start();­

b.start();

a.join();

b.join();

System.**out**.println("Kraj programa");

}

}

**class** ThreadB **extends** ThreadA **implements** Runnable {

**public** ThreadB(String name) {

**super**(name);

}

}

**class** ThreadC **extends** ThreadB {

Thread t;

**public** ThreadC(String name, Thread t) {

**super**(name);

**this**.t = t;

start();

}

**publicvoid** run() {

**try** {

t.join();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**for** (**int** i = 0; i < 10; i++) {

System.**out**.println(name + "1: " + i);

}

}

}

01.

// ThreadD.java

**publicclass** ThreadD **extends** Thread {

**private** String name;

**public** ThreadD(String name) {

**this**.name = name;

}

**publicvoid** run() {

Runnable r = **new** Runnable() {

**publicvoid** run() {

**for**(**int** i=0; i<10; i++){

System.out.println(name + "1: " + i);

}

}

};

**new** Thread(r).start();

**for**(**int** i=0; i<10; i++){

System.out.println(name + "2: " + i);

}

}

**publicstaticvoid** main(String args[]){

ThreadD a = **new** ThreadD("A");

ThreadD b = **new** ThreadD("B");

a.start();

b.start();

}

}

02.

// ThreadE.java

**publicclass** ThreadE **extends** Thread {

**private** String name;

**public** ThreadE(String name) {

**this**.name = name;

}

**publicvoid** run() {

**new** Runnable() {

**publicvoid** run() {

**for**(**int** i=0; i<10; i++){

System.out.println(name + "1: " + i);

}

}

}.start();

**for** (**int** i = 0; i < 10; i++){

System.out.println(name + "2: " + i);

}

}

**publicstaticvoid** main(String args[]){

ThreadE a = **new** ThreadE("A");

ThreadE b = **new** ThreadE("B");

a.start();

b.start();

}

}

**publicclass** ThreadE **extends** Thread {

**private** String name;

**public** ThreadE(String name) {

**this**.name = name;

}

**publicvoid** run() {

**new** Runnable() {

**publicvoid** run() {

**for**(**int** i=0; i<10; i++){

System.out.println(name + "1: " + i);

}

}

}.start();

**for** (**int** i = 0; i < 10; i++){

System.out.println(name + "2: " + i);

}

}

**publicstaticvoid** main(String args[]){

ThreadE a = **new** ThreadE("A");

ThreadE b = **new** ThreadE("B");

a.start();

b.start();

}

03.

// Peta.java

**interface** PrviI **extends** DrugiI, TreciI {

**void** metoda();

}

**interface** DrugiI {

**void** metoda();

}

**interface** TreciI {

**void** metodaI3();

}

**class** Peta **implements** PrviI, TreciI {

**publicvoid** metoda() {

System**.out.**println("metoda");

}

**publicvoid** metodaI3() {

System**.out.**println("metoda I3");

}

}

**class** Dvanaesta {

**publicstaticvoid** main(String [] args) {

Peta e = **new** Peta();

PrviI i = **new** Peta();

DrugiI i2 = e;

i.metoda();

e.metodaI3();

i.metodaI3();

i2.metoda();

}

}

04.

// H.java

**package** paketA;

**interface** I {

**void** metoda();

}

**class** H **implements** I {

**static** H h = **new** H();

**publicstaticvoid** main(String[] args) {

H h = **new** H();

h.metoda();

}

**void** metoda() {

System**.out.**println("abcdef");

}

}

05.

// A.java

**package** paketA;

**class** A {

**staticint** i;

A() {

++i;

}

**privateint** metoda() {

**return** ++i;

}

};

**class** B **extends** A {

B() {

i++;

}

**int** metoda() {

**return** (i + 3);

}

};

**class** X **extends** B {

**publicstaticvoid** main(String ka[]) {

X x = **new** X();

A a = **new** A();

a = (A) x;

System**.out.**println(a.metoda());

}

};

06.

// A.java

**publicclass** A {

**int** j = 1;

**publicstaticvoid** main(String args[]){

metoda();

A a = **new** A();

a.metoda();

}

**publicstaticvoid** metoda(){

**char** digit = 'a';

**for** (**int** i = 0; i < 10; i++){

**switch** (digit){

**case** 'x':

{

**int** j = 0;

System**.out.**println(j);

}

**default**:

{

**int** j = 100;

System**.out.**println(j);

}

}

}

**int** i = j;

System**.out.**println(i);

}

}

07.

// B.java

**publicclass** B {

**int** x = 0, y = 0;

B(**int** a, **int** b){

x = a;

y = b;

}

**protectedint** zbir(){

**return** x + y;

}

**protectedint** razlika(){

**return** x - y;

}

**publicstaticvoid** main(String s[]){

B b = **new** B(1,2);

C c = **new** C();

System**.out.**println(b.razlika());

System**.out.**println(c.razlika());

}

}

**class** C **extends** B{

**publicint** zbir(){

**return** y+x;

}

**publicint** razlika(){

**return** y-x;

}

}

08.

// D.java

**publicclass** D **implements** I3{

**publicvoid** metoda3() {

System**.out.**println(3);

}

**publicvoid** metoda() {

System**.out.**println(1);

}

**publicvoid** metoda2() {

System**.out.**println(2);

}

**publicstaticvoid** main(String args[]){

D d = **new** D();

E e = **new** E();

d.metoda();

e.metoda();

}

}

**class** E **implements** I{

**publicvoid** metoda() {

System**.out.**println(11);

}

}

**interface** I{

**void** metoda();

}

**abstractinterface** I2{

**abstractvoid** metoda2();

}

**interface** I3 **extends** I, I2{

**void** metoda3();

}

09.

//F.java

**publicclass** F {

**privateint** i = 0;

**privatelong** y = 0;

**long** metoda(){

**return** i + y;

}

F napravi(){

**returnnew** F();

}

**publicstaticvoid** main(String args[]){

F f = **new** F();

G g = **new** G();

F f1 = f.napravi();

F f2 = g.napravi();

System**.out.**println(f1.metoda());

System**.out.**println(f2.metoda());

}

}

**class** G **extends** F{

**int** i = 1;

**long** z = 1;

**protectedlong** metoda(){

z = **super**.metoda();

**return** i + z;

}

G napravi(){

**returnnew** G();

}

}

10.

// Klasa0.java

**publicclass** Klasa0 {

**static** {

**int** x = 5;

}

**staticint** x, y;

**publicstaticvoid** main(String args[]) {

x--;

System**.out.**println(x + " " + y);

metoda();

System**.out.**println(x + " " + y);

System**.out.**println(++x + x++);

System**.out.**println(++Klasa0.x);

}

**publicstatic** void metoda() {

y = ++x;

}

}

11.

// Klasa1.java

**publicclass** Klasa1 {

**int** i = 0;

**publicstaticvoid** main(String argv[]) {

}

Klasa1() {

top: **while** (i < 2) {

System**.out.**println(i);

i++;

**continue** top;

}

}

}

12.

// Klasa2.java

**class** Klasa2{

**staticdouble** i = 1;

**staticint** j = 2;

**int** x = 3;

**staticint** y = 6;

**publicstaticvoid** main(String args[]){

metoda();

System**.out.**println(i + j);

System**.out.**println(x + j);

metoda();

System**.out.**println(i + y);

System**.out.**println(x + j);

}

**publicstaticint** metoda(){

**return** (int)i + --y + (j++);

}

**publicstaticdouble** metoda2() {

**return** j++ + --i;

}

};

;;

13.

// Klasa3.java

**publicclass** Klasa3{

**staticint** x = 3;

**publicstatic** void main(String[] args) {

**new** Klasa3();

}

Klasa3() {

Klasa3(2);

}

Klasa3(**int** x) {

System**.out.**println(x);

}

14.

// Klasa4.java

**publicclass** Klasa4 {

**publicstaticvoid** main(String[] args) {

Klasa4 e = **new** Klasa4();

Klasa5 f = **new** Klasa5();

**try** {

f.metoda();

e.metoda();

} **catch** (Exception t) {

System**.out.**println("catch 1");

} **finally** {

System**.out.**println("finally");

}

}

**void** metoda() **throws** CE1 {

**thrownew** CE2("Error 2");

}

}

**class** Klasa5 **extends** Klasa4 {

**void** metoda(){

**try**{

**thrownew** CE1();

} **catch** (CE1 e) {

System**.out.**println("catch 2");

}

}

}

**class** CE1 **extends** Exception {

**public** CE1() {

System**.out.**println("CE1 - 1");

}

**public** CE1(String s) {

System**.out.**println(s);

}

}

**class** CE2 **extends** CE1 {

**public** CE2() {

System**.out.**println("CE2 - 1");

}

**public** CE2(String s) {

System**.out.**println("CE2 - 2");

}

}

15.

// Klasa6.java

**publicclass** Klasa6 **extends** Thread{

**public** Klasa6() {

System**.out.**println("Klasa6()");

}

**publicstatic** void main(String x[]) {

**new** Klasa6().start();

}

**publicvoid** run() {

System**.out.**println("first");

Thread2 niz[] = {**new** Thread3(1), **new** Thread2(), **new** Thread3(3)};

**for** (Thread2 e : niz) {

**if** (e **instanceof** Thread3)

**new** Thread(e).start();

**else**{

**try {**

e.start();

e.join();

} **catch** (InterruptedException e1) {

e1.printStackTrace();

}

}

}

System**.out.**println("last");

}

}

**class** Thread2 **extends** Thread {

**staticint** c = 10;

**int** id;

public Thread2() {

**this**(0);

}

Thread2(int id) {

System**.out.**println("Thread2()");

**this**.id = (id>0)?id:c++;

}

**publicvoid** run() {

**for**(**int** i = 1; i < 6; i++) {

**try** {

sleep(10);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

System**.out.**println("Thread2 - " + id + ": " + i);

}

}

}

**class** Thread3 **extends** Thread2 **implements** Runnable {

Thread3(int id) {

**super**(id);

System**.out.**println("Thread3()");

}

}

16.

// Klasa7.java

**class** A1 {

**private** A1 a1;

**public** A1() {

System**.out.**println("A1");

}

**public** A1(A1 a1){

System**.out.**println("A1(A1)");

**this**.a1 = a1;

}

**void** metoda(){

System**.out.**println("metoda A1");

}

}

**class** A2 **extends** A1 {

A1 a1;

**public** A2() {

**this**(**new** A1());

System**.out.**println("A2(A1)");

}

**public** A2(A1 a1) {

**this**.a1 = a1;

System**.out.**println("A2(A1)");

}

**privatevoid** metoda2(){

System**.out.**println("metoda A2");

}

}

**class** A3 {

**public** A3() {

System**.out.**println("A3");

}

}

**class** Klasa7 **extends** A3 {

**private** A1 a = **new** A2();

**private** A2 a2 = **new** A2(**new** A1());

**public** Klasa7() {

a2.metoda();

System**.out.**println("A4");

a.metoda();

} class A1

**publicstaticvoid** main(String[] args) {

Klasa7 a4 = **new** Klasa7();

a4.metoda();

}

**protectedvoid** metoda(){

System**.out.**println("metoda Klasa7");

}

}

17.

//Clidlet.java

**class** Clidder {

**privatefinalvoid** flipper() { System**.out.**println("Clidder"); }

}

**publicclass** Clidlet **extends** Clidder {

**publicfinalvoid** flipper(){ System**.out.**println("Clidlet"); }

**publicstaticvoid** main(String [] args) {

**new** Clidlet().flipper();

}

}

18.

//DogShow.java

**class** Dog {

**publicvoid** bark() { System**.out.**print("woof "); }

}

**class** Hound **extends** Dog {

**publicvoid** sniff() { System**.out.**print("sniff "); }

**publicvoid** bark() { System**.out.**print("howl "); }

}

**publicclass** DogShow {

**publicstaticvoid** main(String[] args) { **new** DogShow().go(); }

**void** go() {

**new** Hound().bark();

((Dog) **new** Hound()).bark();

((Dog) **new** Hound()).sniff();

}

}

19.

// Hawk.java

**class** Bird {

{ System**.out.**print("b1 "); }

**public** Bird() { System**.out.**print("b2 "); }

}

**class** Raptor **extends** Bird {

**static** { System**.out.**print("r1 "); }

**public** Raptor() { System**.out.**print("r2 "); }

{ System**.out.**print("r3 "); }

**static** { System**.out.**print("r4 "); }

}

**class** Hawk **extends** Raptor {

**static** Bird b = **new** Bird();

**publicstatic** void main(String[] args) {

System**.out.**print("pre ");

**new** Hawk();

System**.out.**println("hawk ");

}

}

20.

// House.java

**class** Building {

Building() {

System**.out.**print("b ");

}

Building(String name) {

**this**();

System**.out.**print("bn " + name);

}

}

**publicclass** House **extends** Building {

House() {

System**.out.**print("h ");

}

House(String name) {

**this**();

System**.out.**print("hn " + name);

}

**publicstaticvoid** main(String[] args) {

**new** House("x ");

}

}

21.

// Fizz.java

**class** Fizz {

**int** x = 5;

**publicstaticvoid** main(String[] args) {

**final** Fizz f1 = **new** Fizz();

Fizz f2 = **new** Fizz();

Fizz f3 = FizzSwitch(f1,f2);

System**.out.**println((f1 == f3) + " " + (f1.x == f3.x));

}

**static** Fizz FizzSwitch(Fizz x, Fizz y) {

**final** Fizz z = x;

z.x = 6;

**return** z;

}

}

23.

// Switch.java

**publicclass** Switch {

**staticint** x = 7;

**publicstaticvoid** main(String[] args) {

String s = "";

**for**(**int** y = 0; y < 3; y++) {

x++;

**switch**(x) {

**case** 8: s += "8 ";

**case** 9: s += "9 ";

**case** 10: { s+= "10 "; **break**; }

**default**: s += "d ";

**case** 13: s+= "13 ";

}

}

System**.out.**println(s);

}

**static** { x++; }

}

24.

// CardPlayer.java

**import** java.io.\*;

**class** Player {

Player(String arg) {

System**.out.**println("parent");

}

}

**class** CardPlayer **extends** Player **implements** Serializable {

**transient** Gadget gadget = **new** Gadget();

CardPlayer() {

**super**("s");

System**.out.**println("child");

}

**publicstaticvoid** main(String[] args) {

CardPlayer c1 = **new** CardPlayer();

**try** {

FileOutputStream fos = **new** FileOutputStream("play.txt");

ObjectOutputStream os = **new** ObjectOutputStream(fos);

os.writeObject(c1);

os.close();

FileInputStream fis = **new** FileInputStream("play.txt");

ObjectInputStream is = **new** ObjectInputStream(fis);

CardPlayer c2 = (CardPlayer) is.readObject();

is.close();

System**.out.**println(c2.gadget);

} **catch** (Exception x ) {

x.printStackTrace();

}

}

}

**class** Gadget {

Gadget() {

System**.out.**println("Gadget created...");

}

}

25.

// TestSer.java

**import** java.io.\*;

**publicclass** TestSer {

**publicstaticvoid** main(String[] args) {

SpecialSerial s = **new** SpecialSerial();

**try** {

ObjectOutputStream oos =

**new** ObjectOutputStream(

**new** FileOutputStream("myFile"));

oos.writeObject(s);

oos.close();

System.**out**.print(++s.z + " ");

ObjectInputStream ois =

**new** ObjectInputStream(

**new** FileInputStream("myFile"));

SpecialSerial s2 = (SpecialSerial) ois.readObject();

ois.close();

System**.out.**println(s2.y + " " + s2.z);

} **catch** (Exception x) {

System**.out.**println("exc");

}

}

}

**class** SpecialSerial **implements** Serializable {

**transientint** y = 7;

**staticint** z = 9;

}

28.

// A1.java

**import** java.io.Serializable;

**publicclass** A1 {

**private** A1 a1;

**public** A1() {

System**.out.**println("A1");

}

**public** A1(A1 a1) {

System**.out.**println("A1(a1)");

**this**.a1 = a1;

}

**publicstaticvoid** main(String args[]) {

**new** A4();

}

**void** metoda(){

System**.out.**println("metoda A1");

}

}

**class** A2 **extends** A1{

A1 a1;

**public** A2() {

**this**(**new** A1());

System**.out.**println("A2");

}

**public** A2(A1 a1) {

**this**.a1 = a1;

System**.out.**println("A2(a1)");

}

}

**class** A3 **extends** A2 **implements** Serializable {

public A3(){

System**.out.**println("A3");

}

}

**class** A4 **extends** A3 {

**private** A1 a = **new** A2();

**private** A2 a2 = **new** A2(**new** A1(**null**));

Serializable a3 = **new** A3();

**public** A4() {

**super**();

System**.out.**println("A4");

a.metoda();

}

}

29.

// A.java

**publicclass** A {

**static** {

**int** a = 5;

System**.out.**println(a);

}

**staticint** a, b;

**publicstaticvoid** main(String args[]) {

a--;

metoda();

System**.out.**println(a + b + ++a);

System**.out.**println(++A.a);

System**.out.**println(a);

}

**publicstaticvoid** metoda() { b = a++ + ++a; }

}

30.

// B1.java

**publicclass** B1 {

**publicvoid** metoda1(){ System**.out.**println("B1 metoda 1"); }

**publicvoid** metoda2(){ System**.out.**println("B1 metoda 2"); }

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

**publicstaticvoid** main(String args[]){

B1 b1 = **new** B1();

B1 b2 = **new** B2() {

public void metoda1() {

System.**out**.println("B2 metoda 1");

}

};

BI b3 = **new** B2(){

**publicvoid** metoda1(){ System**.out.**println("B2 m12"); }

**publicvoid** metoda2(){ System**.out.**println("B2 m22"); }

};

B1 niz[] = {b1, b2, (B1)b3};

for(B1 b : niz){

b.metoda1();

b.metoda2();

}

}

}

**abstractclass** B2 **extends** B1 **implements** BI{

**publicvoid** metoda2(){ System**.out.**println("B2 metoda 2"); }

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

}

**interface** BI {

**abstractvoid** metoda1();

**void** metoda2();

}

31.

// C.java

**publicclass** C {

**int** i = 0;

**publicstaticvoid** main(String args[]) {

C c = **new** C();

}

C() {

**while**(i < 2) {

System**.out.**println(i);

i++;

**continue**;

}

}

}

32.

// E.java

class E {

public static void main(String args[]){

D d = new D();

I1 i = new D();

I2 i2 = d;

i.metoda();

d.metodaI3();

i.metodaI3();

i2.metoda();

}

}

interface I1 extends I2, I3 { void metoda(); }

interface I2{ void metoda(); }

interface I3{ void metodaI3(); }

class D implements I1,I3 {

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

public void metodaI3(){ System.out.println("metoda I3"); }

}

33.

// F.java

public class F{

static int x = 3;

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

F(){ this(2); }

F(int x){

System.out.println(x);

}

}

34.

// G1.java

public class G1{

int i;

{

System.out.println("blok");

System.out.println(i = 1);

}

protected void test(){

System.out.println(i);

System.out.println("iz G1");

}

protected void metoda() throws Exception{

System.out.println("G1 metoda");

test();

}

public static void main(String args[]) throws Exception{

G1 g1 = new G1(), g2 = new G3();

g1.metoda();

g2.metoda();

}

}

abstract class G2 extends G1{

abstract protected void metoda() throws Exception;

protected void test(){

System.out.println(i);

System.out.println("iz G2");

}

}

final class G3 extends G2{

public void metoda() throws Exception {

System.out.println("G3 metoda");

i += 10.51f;

test();

}

}

35.

// L.java

**publicclass** L **extends** K {

**publicstaticvoid** main(String args[]) {

L l = **new** L();

J j = **new** J();

System**.out.**println(J.i);

j = (K) l;

System**.out.**println(j.metoda());

}

};

**class** J {

**staticint** i;

J() { ++i; }

**privateint** metoda() { **return** ++i; }

};

**class** K **extends** J {

**int** i = 0;

K() { i++; }

**int** metoda() { **return** (i+3); }

};

36.

// Klasa5.java

**publicclass** Klasa5 {

**int** x = 0, y = 0;

Klasa5(**int** a, **int** b){

x = a;

y = b;

}

**protectedint** zbir(){

**return** x + y;

}

**protectedint** razlika(){

**return** x - y;

}

**publicstaticvoid** main(String s[]){

Klasa5 b = **new** Klasa5(1,2);

Klasa6 c = **new** Klasa6();

System.**out**.println(b.razlika());

System.**out**.println(c.razlika());

}

}

**class** Klasa6 **extends** Klasa5{

**publicint** zbir(){

**return** y+x;

}

**publicint** razlika(){

**return** y-x;

}

}

38.

// A1.java

**publicclass** A1 **extends** A2 {

**publicstaticvoid** main(String args[]) {

A1 a1 = **new** A1();

A2 a2 = **new** A2();

a1.metoda();

}

**publicvoid** metoda(){

**super**.metoda();

}

}

**class** A2 **extends** A3 {

**public** A2(){

System**.out.**println("A2()");

}

**publicvoid** 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;

}

**publicvoid** metoda() { System**.out.**println(a + b++); }

}

39.

// B1.java

**publicabstractclass** B1 {

B1() {

**super**();

System**.out.**println("B1()");

}

**publicabstractvoid** redefinisi() {}

**publicstaticvoid** main(String args[]) {

B3 b3 = new B3();

b3.metoda();

B2 b2 = b3;

b2.metoda();

B1 b1 = b2;

b1.metoda();

}

**privatevoid** metoda() { System**.out.**println("B1 metoda"); }

}

**abstractclass** B2 **extends** B1 {

B2() {

System**.out.**println("B2()");

}

**abstractprotectedvoid** metoda();

**void** metoda2() { System**.out.**println("B2 metoda"); }

}

**finalclass** B3 **extends** B2 {

B3() {

**super**();

System**.out.**println("B3()");

}

**publicvoid** metoda(){ System**.out.**println("B3 metoda"); }

}

40.Napisati izlaz sljedećeg programa za slučaj kada:

a)Klasa *C2* nasljeđuje klasu *C1*,

b) Klasa*C3* nasljeđuje klasu *C2* i

c) Kada klase *C2* i *C3* ne nasljeđuju druge klase.

**Napomena:** Pretpostaviti da datoteka *proba.txt* postoji na fajl sistemu.

// C1.java

**import** java.io.\*;

**publicclass** C1 {

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

**publicstaticvoid** main(String args[]) **throws** IOException {

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);

} **finally** {

System**.out.**println("Finally iz maina");

}

**try** (BufferedReader bf =

**new** BufferedReader(**new** FileReader("proba.txt"));)

{

**thrownew** IOException();

}

}

**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()");

}

**protectedvoid** metoda() **throws** CE1 {

System**.out.**println("C3 : metoda()");

**thrownew** CE2("CCCCEEEE2");

}

}

**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);

}

}

41.

// D1.java

**publicclass** D1 **extends** D3 **implements** DI {

**publicstaticvoid** main(String args[]) {

D3 niz[] = {**new** D3(), **new** D2(), **new** D1()};

**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()");

**returnnew** D2();

}

}

**class** D3{

**public** D3 metoda() {

System**.out.**println("D3: metoda()");

**returnnew** D3();

}

}

**interface** DI{ D3 metoda(); }

42.

// E1.java

**publicclass** E1 {

**staticpublicvoid** 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");

}

**publicvoid** run() {

**for** (**int** i = 0; i < 6; i++)

System**.out.**println("E2 run");

}

**publicsynchronizedvoid** start() {

**super**.start();

**new** Thread(e3).start();

}

}

**class** E3 **implements** Runnable {

**public** E3() { System**.out.**println("E3"); }

**publicvoid** run() {

**for** (**int** i = 0; i < 6; i++)

System**.out.**println("E3 run");

}

}

43.

// C1.java

**publicclass** C1 {

**publicstaticvoid** main(String args[]) {

C1 c1 = **new** C1();

C2 c2 = **new** C2();

**try** {

System**.out.**println(c1.metoda(c1));

System**.out.**println(c1.metoda(c2));

System**.out.**println(c2.metoda(c1));

System**.out.**println(c2.metoda(c2));

} **catch**(CE1 e) {

System**.out.**println("exception 1");

} **finally** {

System**.out.**println("finally");

}

}

**Object** metoda(C1 c) **throws** CE1 {

**if** (c **instanceof** C1) {

System**.out.**println("metoda");

} **else** {

**thrownew** CE2();

}

**return** 1;

}

}

**class** C2 **extends** C1 {

Object metoda (C1 c) **throws** CE1 {

if( errorCheck() && c **instanceof** C2) {

throw **new** CE2("error 2");

} **elseif** (c **instanceof** C2) {

**thrownew** CE1();

} **else** {

**returnnew** String("abc");

}

}

**boolean** errorCheck(){ **returntrue**; }

}

**class** CE1 **extends** Throwable {

**public** CE1() { System**.out.**println("ce1 - 1"); }

**public** CE1(String s) {

**super**(s);

System**.out.**println("ce1 - 2");

}

}

**class** CE2 **extends** RuntimeException {

**public** CE2() {

System**.out.**println("ce2 - 1");

}

**public** CE2(String s) {

**this**();

System**.out.**println("ce2 - 2");

}

}

44.

// E1.java

**publicclass** E1 **extends** Thread {

**private** String name;

**public** E1(String name) {

**this**.name = name;

}

**publicvoid** run() {

Runnable r = **new** Runnable() {

**publicvoid** run(){

**for**(**int** i = 0; i < 5; i++) {

System**.out.**println(i);

}

}

};

**new** Thread(r).start();

**synchronized**(**this**) {

**for**(**int** i = 0; i < 100; i++) {

System**.out.**println(i + " " + **this**.name);

}

}

}

**publicstaticvoid** main(String args[]) {

E1 a = **new** E1("A");

E1 b = **new** E1("B");

a.start();

b.start();

}

}

45.Napisati izlaz sljedećeg programa. Šta će se desiti ako u klasi *D1* odkomentarišemo metodu *metoda()* a zakomentarišemo metodu *metoda(int i)*?

// D1.java

**publicclass** D1 **extends** DI1.D2 **implements** DI1, DI2 {

**public** D1() {

**super**();

System**.out.**println("D1()");

}

**publicstaticvoid** main(String args[]) {

DI2 di1 = **new** D1();

DI2 di2 = **new** D2();

DI1 di3 = **new** D1();

D1 d1 = **new** D1();

D2 d2 = **new** DI1.D2();

System**.out.**println(((DI2)**new** D1()).metoda());

System**.out.**println(((DI2)d2).metoda());

}

// public void metoda() {

// System**.out.**println("D1 metoda");

// }

public int metoda(int i) {

System**.out.**println("D1 metoda");

return 1;

}

}

**interface** DI1 {

**class** D2 **implements** DI2 {

D2() {

System**.out.**println("D2()");

}

**publicint** metoda() {

System**.out.**println("D2 metoda");

**return** 0;

}

}

}

**interface** DI2 {

**int** metoda();

}

46.Ukoliko postoji greska napisati gdje se nalazi, pa ispisati izlaz programa u slučaju da se zakomentariše linija sa greškom.

// F1.java

**publicclass** F1 {

**staticboolean** b;

**staticint** counter = 0;

**publicstaticvoid** main(String args[]) {

**try**{

label:

**while**(counter == 0){

**if**(!b){

System**.out.**println("C");

b = **true**;

main(new String[]{"D"});

System**.out.**println("E");

} **else** {

counter++;

System**.out.**println("A");

System**.out.**println(args[0]);

main((String[])**new** Object("F"));

**if**(counter == 1) {

**continue** label;

System**.out.**println("B");

}

}

System**.out.**println(F1.getF1());

}

} **catch**(Exception e) {

System**.out.**println("exception");

}

}

**publicstatic** F1 getF1() {

System**.out.**println("1");

**returnnew** F1();

}

{

System**.out.**println("F1");

}

}

47.

// G1.java

**import** java.io.\*;

**publicclass** G1 {

**publicstaticvoid** 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;

**transientint** b = 2;

**public** G2() {

System**.out.**println("G2 konstruktor");

}

**publicvoid** writeExternal(ObjectOutput out)

**throws** IOException {

out.write(3);

out.write(4);

System**.out.**println("G2 writeExternal");

}

**publicvoid** readExternal(ObjectInput in)

**throws** IOException, ClassNotFoundException {

System**.out.**println("G2 readExternal");

}

}

**class** G3 **implements** Serializable {

**int** a = 5;

**transientint** b = 6;

**public** G3(String s){

System**.out.**println("G3 konstruktor");

}

**privatevoid** writeObject(ObjectOutputStream out)

**throws** IOException {

System**.out.**println("G3 writeObject");

out.write(a);

out.write(b);

}

**privatevoid** readObject(ObjectInputStream in)

**throws** IOException, ClassNotFoundException {

System**.out.**println("G3 readObject");

a = in.read();

b = in.read();

}

}

48.

// A.java

**publicclass** A {

**publicstaticvoid** main(String[] args) {

String a = "newspaper";

a = a.substring(5,7);

System.**out**.println(a);

**char** b = a.charAt(1);

System.**out**.println(b);

a = a + b;

System.**out**.println(a);

}

}

49.

// A.java

public class A{

public static void main(String[] args) {

String a = "newspaper";

a = a.substring(5,7);

System.out.println(a);

char b = a.charAt(1);

System.out.println(b);

a = a + b;

System.out.println(a);

C c = new C();

c.ispis();

c.metoda();

}

}

abstract class B extends A {

protected abstract void metoda();

protected String ispis(){ return "ispis"; }

}

class C extends B{

private void metoda(){ System.out.println("C"); }

}

50.

// HelloWorldAnonymousClasses.java

public class HelloWorldAnonymousClasses {

interface HelloWorld {

public void greet();

public void greetSomeone(String someone);

}

public void sayHello() {

class EnglishGreeting implements HelloWorld {

String name = "world";

public void greet() { greetSomeone("world"); }

public void greetSomeone(String someone) {

name = someone;

System.out.println("Hello " + name);

}

}

HelloWorld englishGreeting = new EnglishGreeting();

HelloWorld frenchGreeting = new HelloWorld() {

String name = "tout le monde";

public void greet() { greetSomeone("tout le monde"); }

public void greetSomeone(String someone) {

name = someone;

System.out.println("Salut " + name);

ispis();

}

public void ispis() { System.out.println("Dodao"); }

};

HelloWorld spanishGreeting = new HelloWorld() {

String name = "mundo";

public void greet() { greetSomeone("mundo"); }

public void greetSomeone(String someone) {

name = someone;

System.out.println("Hola, " + name);

}

};

englishGreeting.greet();

frenchGreeting.greetSomeone("Fred");

spanishGreeting.greet();

}

public static void main(String... args) {

HelloWorldAnonymousClasses myApp =

new HelloWorldAnonymousClasses();

myApp.sayHello();

}

}

51.

// AnonimnaKlasa.java

public class AnonimnaKlasa {

public static void main(String args[]){

int i = 1, j = 0;

switch(i){

case 2: j += 6;

case 4: j += 1;

default: j += 2;

case 0: j += 4;

}

System.out.println("j = " + j);

} }

52

// Animal.java

.

public class Animal {

public void eat(){ System.out.println("Animal is eating"); }

public void drink(){ System.out.println("Animal is drinking");}

private void privateMethod() {

System.out.println("Animal's private method");

}

public void commonMethod() { System.out.println("Common method"); }

public static void main(String[] args) {

Animal a = new Cat();

a.eat();

a.drink();

a.privateMethod();

a.commonMethod();

((Cat)a).meow();

Cat c = (Cat)a;

c.eat();

c.drink();

((Animal)c).privateMethod();

c.commonMethod();

c. meow();

Animal animal = new Animal();

Cat catty = (Cat) animal;

}

}

class Cat extends Animal {

public void eat() { System.out.println("Cat is eating"); }

public void drink() { System.out.println("Cat is drinking"); }

public void meow() { System.out.println("Cat is meowing"); }

}

54.

// B1.java

**publicabstractclass** B1 {

B1() {

**super**();

System.**out**.println("B1()");

}

**publicstaticvoid** main(String[] args) {

B3 b3 = **new** B3();

b3.metoda();

B2 b2 = **new** B2();

b2 = b3;

b2.metoda();

B1 b1 = b2;

b1.metoda();

B2 test2 = **new** B2();

test2.metoda();

}

**void** metoda() {

System.**out**.println("B1 metoda");

}

}

**class** B2 **extends** B1 {

B2() {

System.**out**.println("B2()");

}

**protectedvoid** metoda() {

System.**out**.println("B2 metoda");

}

**void** metoda2() {

System.**out**.println("B2 metoda");

}

}

**finalclass** B3 **extends** B2 {

B3() {

**super**();

System.**out**.println("B3()");

}

**publicvoid** metoda() {

System.**out**.println("B3 metoda");

}

}

55.

// Go.java

**publicclass** Go **extends** Game {

Go() {

**super**(s2);

}

{

s += "i ";

}

**publicstaticvoid** main(String[] args) {

**new** Go();

System.**out**.println(s);

}

**static** {

s += "sb ";

}

}

**class** Game {

**static** String s = "-";

String s2 = "s2";

Game(String arg) {

s += arg;

}

}

56.

// LocalInnerClass.java

**interface** Counter {

**int** next();

}

**publicclass** LocalInnerClass {

**privateint** count = 0;

Counter getCounter(**final** String name) {

**class** LocalCounter **implements** Counter {

**public** LocalCounter() {

System.out.println("LocalCounter()");

}

**publicint** next(){

System.out.println(name);

**return** count++;

}

}

**returnnew** LocalCounter();

}

Counter getCounter2(**final** String name) {

**returnnew** Counter() {

{

System.**out**.println("Counter()");

}

**publicint** next() {

System.**out**.println(name);

**return** count++;

}

};

}

**publicstaticvoid** main(String[] args) {

LocalInnerClass lic = **new** LocalInnerClass();

Counter

c1 = lic.getCounter("Local inner"),

c2 = lic.getCounter2("Anonymous inner");

for (**int** i = 0; i < 5; i++)

System.**out**.println(c1.next());

for (**int** i = 0; i < 5; i++)

System.**out**.println(c2.next());

}

}

57.

// TheDiamond.java

**publicclass** TheDiamond {

**publicstaticvoid** main(String[] args) {

MyGenInterface<String> ref = **new** MyGenClass<>("String");

MyGenClass ref2 = (MyGenClass) ref;

System.**out**.println(ref.getT());

System.**out**.println(ref2.getT1());

}

}

**class** MyGenClass<T1,T2>**implements** MyGenInterface<T2> {

**private** T1 t1;

**private** T2 t2;

**public** MyGenClass(T2 t2) {

t1 = (T1) **new** Double("3.14159");

**this**.t2 = t2;

}

**public** T1 getT1() {

**return** t1;

}

@Override

**public** T2 getT() {

**return** t2;

}

}

**interface** MyGenInterface<T> {

T getT();

}

58.

// StrCmp.java

**publicclass** StrCmp {

**publicstaticvoid** main(String ... arg) {

String test1 = "Foo";

String test2 = **new** String("Bar");

String test3 = "Bar";

String test4 = test1 + test3;

System.**out**.println(test2 == test3);

System.**out**.println(test2.equals(test3));

System.**out**.println(test2.compareTo(test3));

System.**out**.println(test4 == "FooBar");

System.**out**.println(test4 == **new** String("FooBar"));

System.**out**.println("FooBar".equals(test4));

System.**out**.println(("Foo" + "Bar") == "FooBar");

}

}

59.

// Players.java

import java.io.\*;

**publicclass** Players {

**publicstaticvoid** main(String[] args)

**throws** IOException, ClassNotFoundException {

System.**out**.println("Constructing objects:");

Player1 b1 = **new** Player1();

Player2 b2 = **new** Player2();

ObjectOutputStream o =

**new** ObjectOutputStream(

**new** FileOutputStream("Players.out"));

System.**out**.println("Saving objects:");

o.writeObject(b1);

o.writeObject(b2);

o.close();

ObjectInputStream in =

**new** ObjectInputStream(

**new** FileInputStream("Players.out"));

System.**out**.println("Recovering b1:");

b1 = (Player1) in.readObject();

System.out.println("Recovering b2:");

b2 = (Player2) in.readObject();

}

}

**class** Player1 **implements** Serializable {

**public** Player1() {

System.**out**.println("Player1 Constructor");

}

**publicvoid** writeExternal(ObjectOutput out) **throws** IOException {

System.**out**.println("Player1.writeExternal");

}

**publicvoid** readExternal(ObjectInput in)

**throws** IOException, ClassNotFoundException {

System.**out**.println("Player1.readExternal");

}

}

**class** Player2 **implements** Externalizable {

**public** Player2() {

System.**out**.println("Player2 Constructor");

}

**publicvoid** writeExternal(ObjectOutput out) **throws** IOException {

System.**out**.println("Player2.writeExternal");

}

**publicvoid** readExternal(ObjectInput in)

**throws** IOException, ClassNotFoundException {

System.**out**.println("Player2.readExternal");

}

}

60.

// BaseClass.java

class BaseClass {

private void foo(){ System.out.println("In BaseClass.foo()"); }

void bar(){ System.out.println("In BaseClass.bar()"); }

public static void main(String[] args) {

DerivedClass po = new DerivedClass();

((BaseClass)po).foo();

((BaseClass)po).bar();

}

}

class DerivedClass extends BaseClass {

void foo(){ System.out.println("In Derived.foo()"); }

void bar(){ System.out.println("In Derived.bar()"); }

}

61.

// KlasaA.java

private class KlasaA {

public static void main(String ... args){

int a = 0;

int b = 5;

System.out.println(a++ + b);

}

}

62.

// Bonds.java

class Bonds {

Bonds force() { return new Bonds(); }

}

public class Covalent extends Bonds {

Covalent force() { return new Covalent(); }

public static void main(String[] args) {

new Covalent().go(new Covalent());

}

void go(Covalent c) {

go2(new Bonds().force(), c.force());

}

void go2(Bonds b, Covalent c) {

Covalent c2 =(Covalent)b;

Bonds b2 = (Bonds)c;

}

}

63.

// Cycles.java

public class Cycles {

public static void rideCycle(CycleFactory factory) {

Cycle c = factory.getCycle();

c.ride();

}

public static void main(String [] args) {

rideCycle(new UnicycleFactory());

rideCycle(new BicycleFactory());

rideCycle(new TricycleFactory());

}

}

interface Cycle { void ride(); }

interface CycleFactory { Cycle getCycle(); }

class Unicycle implements Cycle {

public void ride() {

System.out.println("Ride Unicycle");

}

}

class UnicycleFactory implements CycleFactory {

public Cycle getCycle() {

return new Unicycle();

}

}

class Bicycle implements Cycle {

public void ride() { System.out.println("Ride Bicycle"); }

}

class BicycleFactory implements CycleFactory {

public Cycle getCycle() { return new Bicycle(); }

}

class Tricycle implements Cycle {

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

public void ride() { System.out.println("Ride Tricycle"); }

}

class TricycleFactory implements CycleFactory {

public Cycle getCycle() {

return new Tricycle();

}

}

64.

// Equals.java

class Equals {

public static void main(String [] args){

int x = 100;

double y = 100.1;

boolean b = (x = y);

System.out.println(b);

}

}

65.

// EqualsS.java

class EqualsS{

public static void main(String [] args) {

int x = 97;

char y = 'a';

boolean b = (x == y);

System.out.println(b);

}

}

66.

// Ex11.java

public class Ex11 {

public static void main(String[] args) {

Test t = new Test();

t.f().say("hi");

((Inner)t.f()).say("hello");

}

}

interface Ex11Interface {

void say(String s);

}

class Test {

private class Inner implements Ex11Interface {

public void say(String s) {

System.out.println(s);

}

}

Ex11Interface f() {

return new Inner();

}

}

67.

// Exercise7.java

public class Exercise7 {

private int x;

private void metoda() {

System.out.println("Exercise7.metoda()");

}

private void metoda2() {

Exercise7Inner e7i = new Exercise7Inner();

e7i.metodaModify();

System.out.println(x);

System.out.println(e7i.y);

}

private class Exercise7Inner {

private int y;

private void metodaModify() { x = 5; }

}

public static void main(String[] args) {

Exercise7 e7 = new Exercise7();

e7.metoda2();

}

}

68.

// Testiranje.java

public class Testiranje{

public static void main(String args[]){

try {

throw new Exc1();

}catch (Exc0 e0){

System.out.println("Ex0 caught");

}catch (Exception e){

System.out.println("exception caught");

}

}

}

class Exc0 extends Exception { }

class Exc1 extends Exc0 { }

69.

// G.java

public class G{

static int x = 3;

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

G(){ new G(2); }

G(int x){ System.out.println(x); }

}

70.

// ExamQuestion7.java

public class ExamQuestion7{

static int j;

static void methodA(int i){

boolean b;

do{

b = i<10 | methodB(4);

b = i<10 || methodB(8);

}while (!b);

}

static boolean methodB(int i){

j += i;

return true;

}

public static void main(String[] args){

methodA(0);

System.out.println( "j = " + j );

}

}

71.

// Igra.java

public class Igra {

public static void main(String[] args) {

Integer i = new Integer(5);

Integer j = new Integer(5);

System.out.println(i == j);

System.out.println( i == (new Integer(5))); }

}

72.

// HummingBird.java

public class HummingBird extends Bird {

public static void fly() { s += "hover "; }

public static void main(String[] args) {

Bird b1 = new Bird();

Bird b2 = new HummingBird();

Bird b3 = (HummingBird)b2;

HummingBird b4 = (HummingBird)b2;

b1.fly(); b2.fly(); b3.fly(); b4.fly();

System.out.println(s);

}

}

class Bird {

public static String s = "";

public static void fly() { s += "fly "; }

}

73.

// Izuzetak.java

public class Izuzetak{

public static void main(String[] args) {

try{

int x = 0;

int y = 5 / x;

}

catch (Exception e){

System.out.println("Exception");

}catch (ArithmeticException ae) {

System.out.println(" Arithmetic Exception");

}

System.out.println("finished");

}

}

74.

// MaskiranjeClanova.java

class MaskiranjeClanova {

int x = 5;

}

public class MaskiranjeClanovaDva extends MaskiranjeClanova {

int x = 6;

public static void main(String[] args) {

MaskiranjeClanova mc = new MaskiranjeClanova();

MaskiranjeClanova mcd = new MaskiranjeClanovaDva();

System.out.println(mcd.x + " " + mc.x);

}}

75.

// MyExceptionTest.java

public class MyExceptionTest {

public static void main(String[] args) {

try {

second();

}catch(MyException e) {

System.out.println("Uhvacen i obradjen izuzetak " + e);

}

}

public static void second() throws MyException { first(); }

public static void first() throws MyException {

throw new MyException("Poruka o gresci");

}

}

class MyException extends Exception {

MyException() {

super();

}

MyException(String s) {

super(s);

System.out.println("MyException super");

}

}

76.

// Test.java

class Test {

public static void main(String []args) {

new MyThread("nit");

}

}

class MyThread extends Thread {

public MyThread(String name) {

this.setName(name);

start();

System.out.println("MyThread " + getName());

}

public void start() {

System.out.println("start " + getName());

}

public void run() {

System.out.println("run " + getName());

}

}

77.

// MyThread.java

class MyThread extends Thread {

public void run(){

System.out.println("run - ime niti: " + Thread.currentThread().getName());

}

public static void main(String args[])throws Exception{

Thread myThread = new MyThread();

myThread.start();

MyThread nit = new MyThread();

nit.start();

nit.join();

System.out.println("main - ime niti: " + Thread.currentThread().getName());

}

}

78.

// One.java

class One {

void go1() { System.out.print("1 "); }

final void go2() { System.out.print("2 "); }

private void go3() { System.out.print("3 "); }

}

public class OneB extends One {

void go1() { System.out.print("1b "); }

void go3() { System.out.print("3b "); }

public static void main(String[] args) {

new OneB().go1();

new One().go1();

new OneB().go2();

new OneB().go3();

new One().go3();

}

}

79.

// PassA.java

class PassA {

public static void main(String [] args) {

PassA p = new PassA();

p.start();

}

void start() {

long [] a1 = {3,4,5};

fix(a1);

System.out.println(a1[0] + " " + a1[1] + " " + a1[2]);

}

long [] fix(long [] a3) {

a3[1] = 7;

return a3;

}

}

80.

// PenguinTest.java

public class PenguinTest {

public static void main(String []args) {

Penguin pingu = new Penguin();

pingu.walk();

pingu.fly();

}

}

class CannotFlyException extends Exception {}

interface Birdie {

public abstract void fly() throws CannotFlyException;

}

interface Biped { void walk(); }

abstract class NonFlyer {

private void fly() { System.out.print("cannot fly "); }

}

class Penguin extends NonFlyer implements Birdie, Biped {

public void walk() { System.out.print("walk\n"); }

}

81. OVAJ PONOVO

// A1.java

public class A1{

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

private A1 a1;

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

public A1(A1 a1){

System.out.println("A1(a1)");

this.a1 = a1;

}

public static void main(String args[]){ new A4(); }

public void metoda(){ System.out.println("metoda A1"); }

}

class A2 extends A1{

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

public A1 d = new A1(null);

A1 a1;

public A2(){

this(new A1());

System.out.println("A2");

}

public A2(A1 a1){

this.a1 = a1;

System.out.println("A2(a1)");

}

static A3 as = new A3();

}

class A3 extends A2 implements Serializable{

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

public A1 adas = new A1(null);

static A1 as = new A1(null);

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

}

class A4 extends A3{

private A1 a = new A2();

{

System.out.println("nestaticki A4");

}

static A2 asd = new A2(null);

private A2 a2 = new A2(new A1(null));

Serializable a3 = new A3();

public A4(){

super();

System.out.println("A4");

a.metoda();

}

}