Exceptions

1. Determine whether the following program will generate (i) compilation errors, (ii) runtime errors. If the program does not generate errors, say what it will print out; if the program generates errors, correct them and say what it will print out after the correction. Motivate your answers

public class Exercise1 {

public static void main(String[] args) {

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

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

}

System.out.println("I’ve printed out the sums of i and j up to " + i + "," + j);

}

}

1. Determine whether the following program will generate (i) compilation errors, (ii) runtime errors. If the program does not generate errors, say what it will print out; if the program generates errors, correct them and say what it will print out after the correction. Motivate your answers.

public class Exercise2 {

private int x = 101;

private void f(int x) {

x++; g();

}

private void g() {

System.out.println(x);

}

public static void main(String[] args) {

Exercise2 e = new Exercise2();

int x = 200;

e.f(x);

}

}

1. Determine whether the following classes will generate (i) compilation errors, (ii) runtime errors. If the program does not generate errors, say what it will print out; if the program generates errors, correct them and say what it will print out after the correction. Motivate your answers.

public class Base {

public Base() {

infob = "I am an object of the Base class";

}

public String getInfo() {

return infob;

}

private String infob;

}

public class Derived extends Base {

public Derived() {

super();

infod = "I am an object of the Derived class";

}

public String getInfo() {

return infod + ", " + super.getInfo();

}

private String infod;

}

public class Exercise3 {

public static void main(String[] args) {

Base b = new Base();

Derived d = new Derived();

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

System.out.println(d.getInfo());

b = d;

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

}

}

1. Capture all exceptions in the following program, printing out error messages that describe the type of error that occurred.

import java.io.\*;

public class Exercise4 {

public static void main(String[] args) {

int n=10;

int[] v = new int[n];

FileReader f = new FileReader("dati.txt");

BufferedReader in = new BufferedReader(f);

int i=0;

String linea = in.readLine();

while (linea!=null) {

v[i] = Integer.parseInt(linea);

linea = in.readLine();

i++;

}

f.close();

}

}

1. Solve Exercise 4 (above program) by handling explicitly all exceptions by printing out suitable error messages.
2. Define a new exception, called ExceptionLineTooLong, that prints out the error message "The strings is too long". Write a program that reads all lines of a file and throws an exception of type ExceptionLineTooLong in the case where a string of the file is longer than 80 characters. Handle also all exceptions that could be thrown by the program.
3. Write a class containing the following static methods:

• fileExists, that takes as a parameter a string and returns a boolean value that is true, if

the file whose name is passed as parameter exists, and false otherwise;

• isInt, that takes as parameter a string and returns a boolean value that is true if the

string represents an integer, and false otherwise.

• isDouble, that takes as parameter a string and returns a boolean value that is true if

the string represents a real number, and false otherwise.

Solve the exercise by suitably catching exceptions