OKANAGAN

COSC 121

Computer Programming II

Midterm #1 - Review

Dr. Mostafa Mohamed

Acknowledgement: The slides mainly rely on the textbook of Y. D. Liang titled “Introduction to Java Programming, 10th Ed.”, Pearson Edu. Inc.

Midterm Format

Part 1: Multiple choice questions

Part 2: Code analysis questions

- **For example,**
  - What is the output of a given code?
  - Given some code, what will happen if...?
  - Add/change statements in order for your code to have a certain behavior

Part 3: Programming questions

- **Short / Long coding-questions**

Exceptions

Exception handling

- How it works and exception propagation

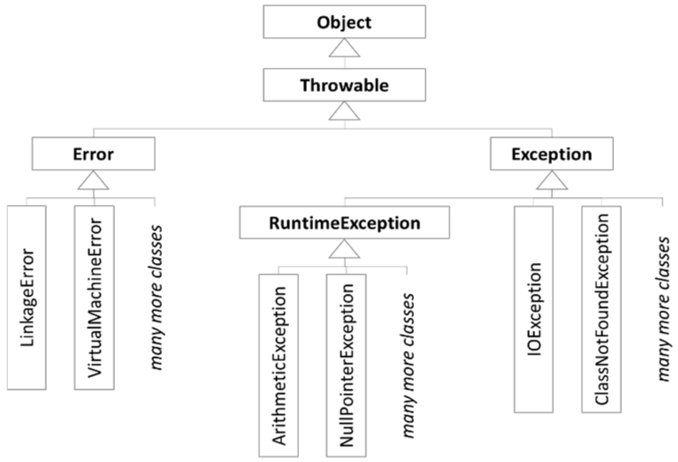
Checked vs. unchecked exceptions.

- Checked exceptions
  - Exception subclasses except
  - RuntimeException. For example:
    - IOException and its subclasses (MalformedURLException, EOFException, FileNotFoundException)
- Unchecked exceptions
  - Include Error and RuntimeException and all their subclasses. For example:
    - ArithmeticException, NullPointerException, IndexOutOfBoundsException, NumberFormatException

Dealing with exceptions: Any exception is either:

- Handled within the current method (try..catch) , or
- Thrown to the calling method (declare in method header)
  - This is the default for unchecked exceptions

Defensive programming vs. exception handling



Java I/O

Java I/O

- 3 steps for using streams: Open stream, read/write, close stream.

Text I/O

- The classes and their methods
  - How to use with standard I/O, files, and web.
  - Scanner, PrintWriter, BufferedReader, BufferedWriter

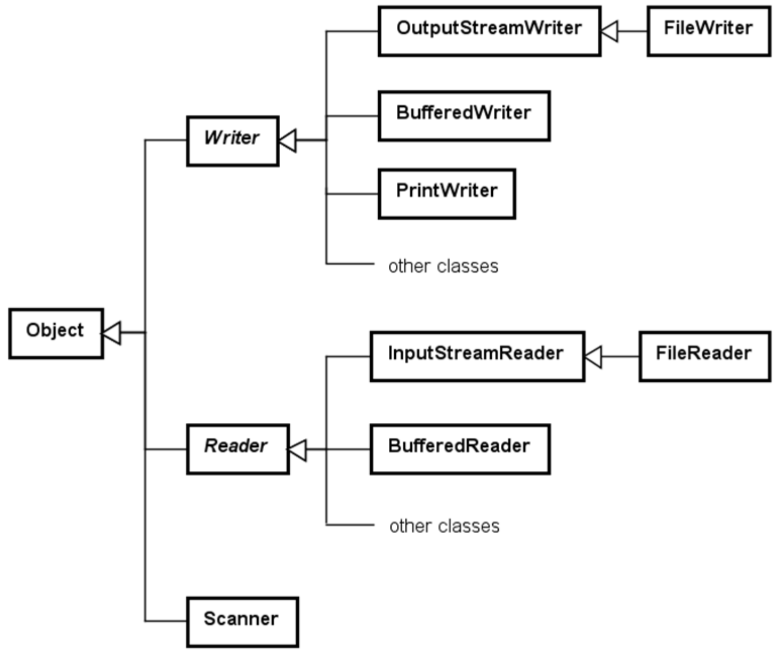
Binary IO

- The classes and their methods
  - Input:
    - FileInputSteam, DataInputStream, ObjectInputStream
    - BufferedInputStream
  - Output
    - FileOutputSteam, DataOutputStream,ObjectOutputStream
    - BufferedOutputStream

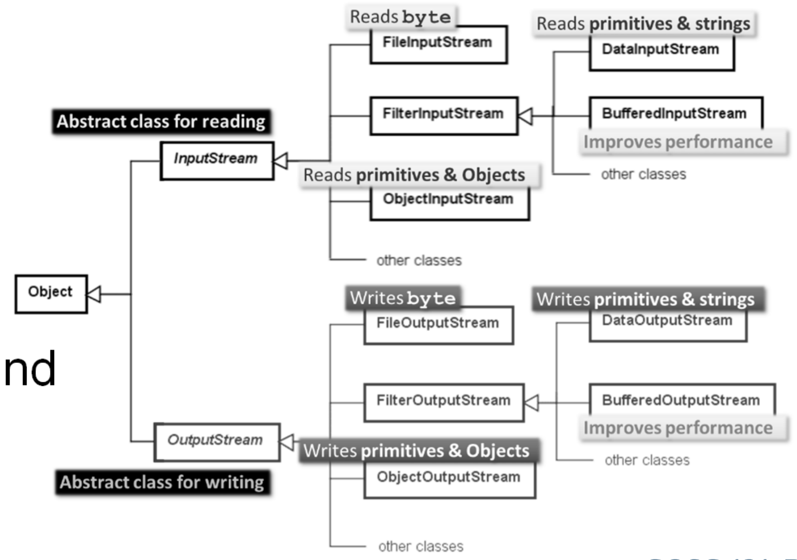
Checking input validity using I/O streams and Exception handling.

File, URL classes

Try-with-resources



nd



The list of examples in the following section is **NOT inclusive.**
In other words, do **not** just focus on these examples and assume the exam will be based on them.

Your preparation for the exam should include **ALL** exercises we worked on in this course (up to the cut-off lecture of this exam) such as those discussed in the lecture notes and in the lab assignments.

Analysis: Exception Handling

Suppose that **statement2 causes an exception**:

```
try {
   statement1;
   statement2;
   statement3;
} catch (Exception1 ex1) {
} catch (Exception2 ex2) {
}
statement4;
```

- Will **statement3** be executed?
- If the exception is not caught, will **statement4** be executed?
- If the exception is caught in the **catch** block, will **statement4** be executed?

Analysis: Solution

Will **statement3** be executed?

- No. the remaining part of the try block is skipped

If the exception is not caught, will **statement4** be executed?

- No. the exception will be thrown to the caller method.

If the exception is caught in the **catch** block, will **statement4** be executed?

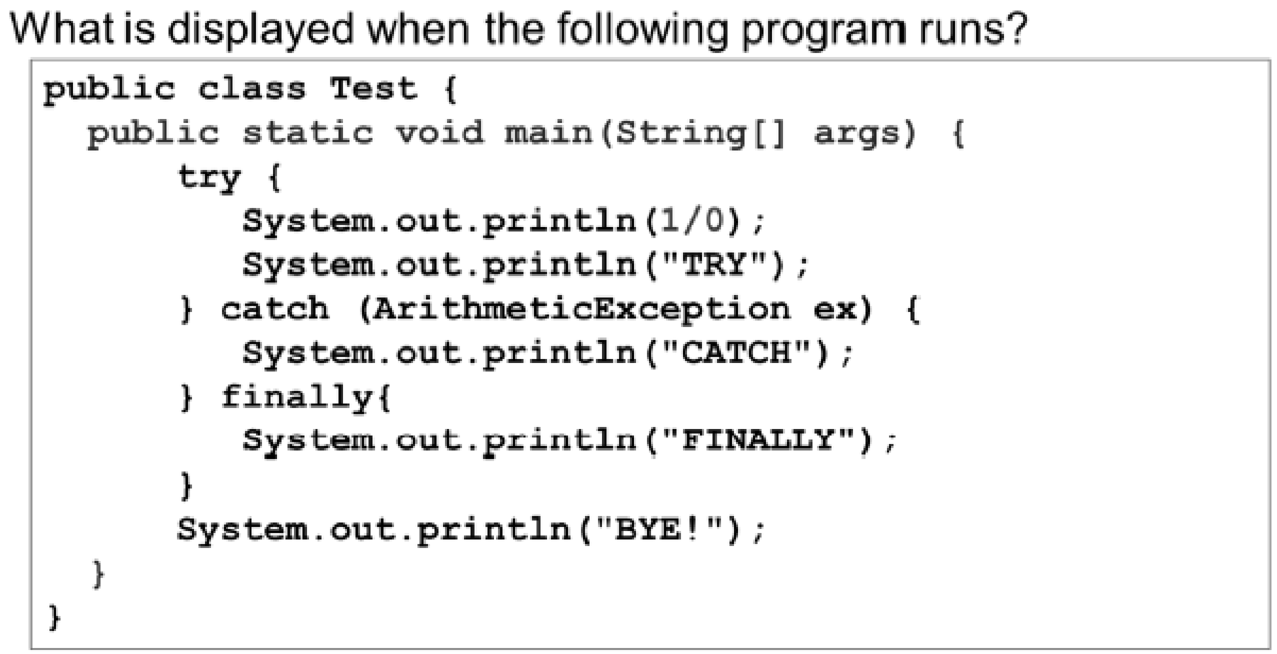
- Yes. Only the remaining part of the try block is skipped (remember that we can have more than one try following each other in the same method.

Analysis Questions: Exception Handling Example

What is displayed when the following program runs?

```
public class Test {
  public static void main(String[] args) {
      try {
         System.out.println(4/2);
         System.out.println("TRY");
      } catch (ArithmeticException ex) {
         System.out.println("CATCH");
      } finally{
         System.out.println("FINALLY");
      }
      System.out.println("BYE!");
  }
}
```

Solution



What is displayed when the following program runs?

```
public class Test {
  public static void main(String[] args) {
      try {
         System.out.println(1/0);
         System.out.println("TRY");
      } catch (ArithmeticException ex) {
         System.out.println("CATCH");
      } finally{
         System.out.println("FINALLY");
      }
      System.out.println("BYE!");
  }
}
```

```
CATCH
FINALLY
BYE!
```

Analysis Questions: Exception Handling Example

What is displayed when the following program runs?

```
public class Test {
  public static void main(String[] args) {
      try {
         System.out.println(1/0);
         System.out.println("TRY");
      } catch (NullPointerException ex) {
         System.out.println("CATCH");
      } finally{
         System.out.println("FINALLY");
      }
      System.out.println("BYE!");
  }
}
```

Solution

What is displayed when the following program runs?

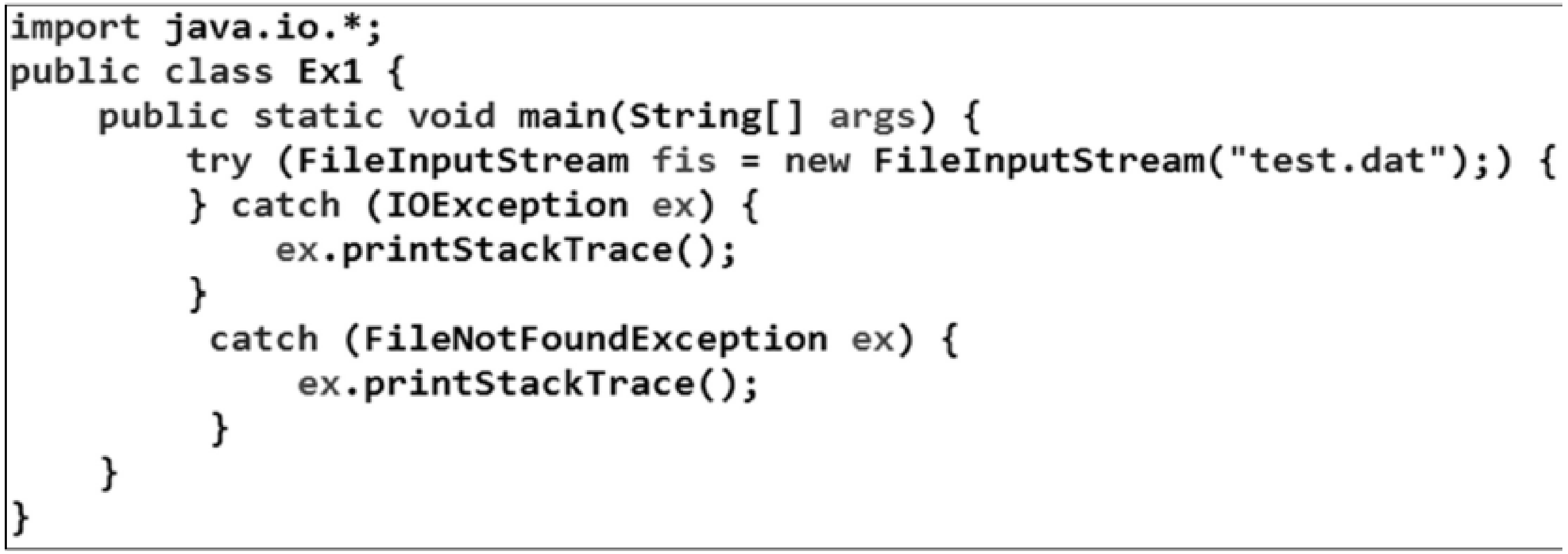
```
public class Test {
  public static void main(String[] args) {
      try {
         System.out.println(1/0);
         System.out.println("TRY");
      } catch (NullPointerException ex) {
         System.out.println("CATCH");
      } finally{
         System.out.println("FINALLY");
      }
      System.out.println("BYE!");
  }
}
```

FINALLY

error message

Analysis Questions: Exception Handling Example

What is wrong ?

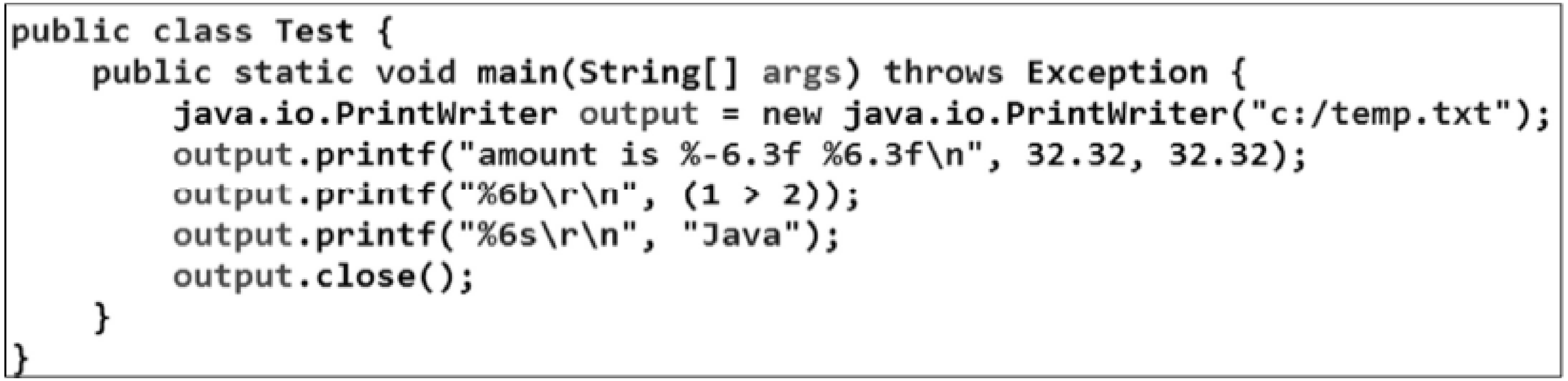


```
import java.io.*;
public class Ex1 {
    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("test.dat");) {
        } catch (IOException ex) {
            ex.printStackTrace();
        }
         catch (FileNotFoundException ex) {
             ex.printStackTrace();
         }
    }
}
```

FileNotFoundException is not reachable.

Analysis Questions: Text I/O Example

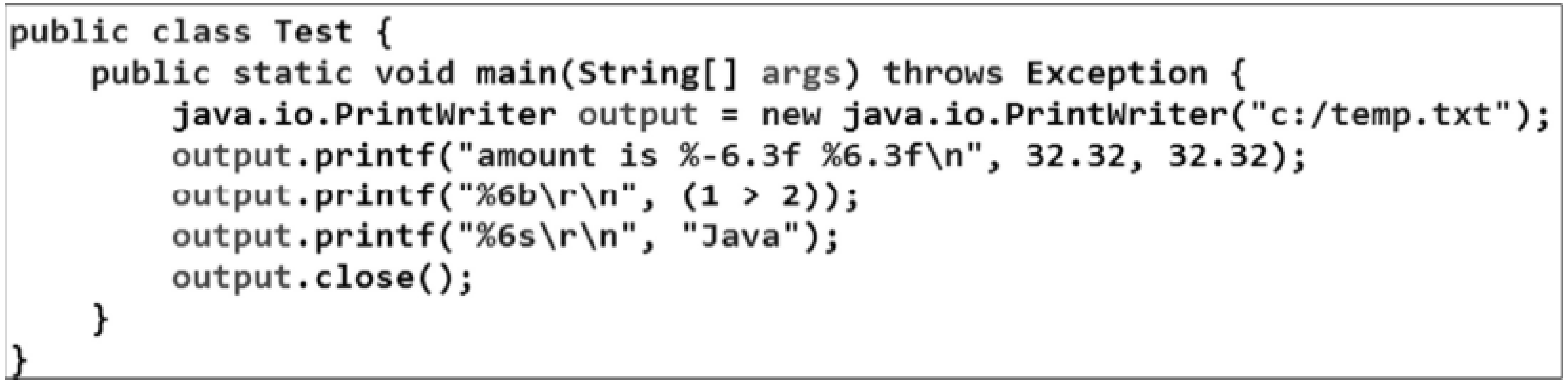
What are the contents of **temp.txt** after this program is executed



```java
public class Test {
    public static void main(String[] args) throws Exception {
        java.io.PrintWriter output = new java.io.PrintWriter("c:/temp.txt");
        output.printf("amount is %-6.3f %6.3f\n", 32.32, 32.32);
        output.printf("%6b\r\n", (1 > 2));
        output.printf("%6s\r\n", "Java");
        output.close();
    }
}
```

Solution

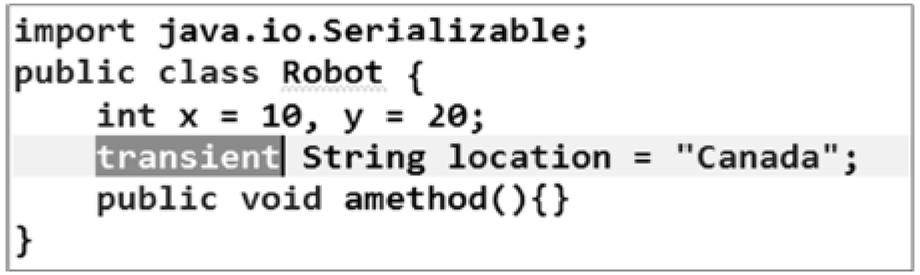
What are the contents of **temp.txt** after this program is executed



```
public class Test {
    public static void main(String[] args) throws Exception {
        java.io.PrintWriter output = new java.io.PrintWriter("c:/temp.txt");
        output.printf("amount is %-6.3f %6.3f\n", 32.32, 32.32);
        output.printf("%6b\r\n", (1 > 2));
        output.printf("%6s\r\n", "Java");
        output.close();
    }
}
```

```
amount is 32.320 32.320
 false
  Java
```

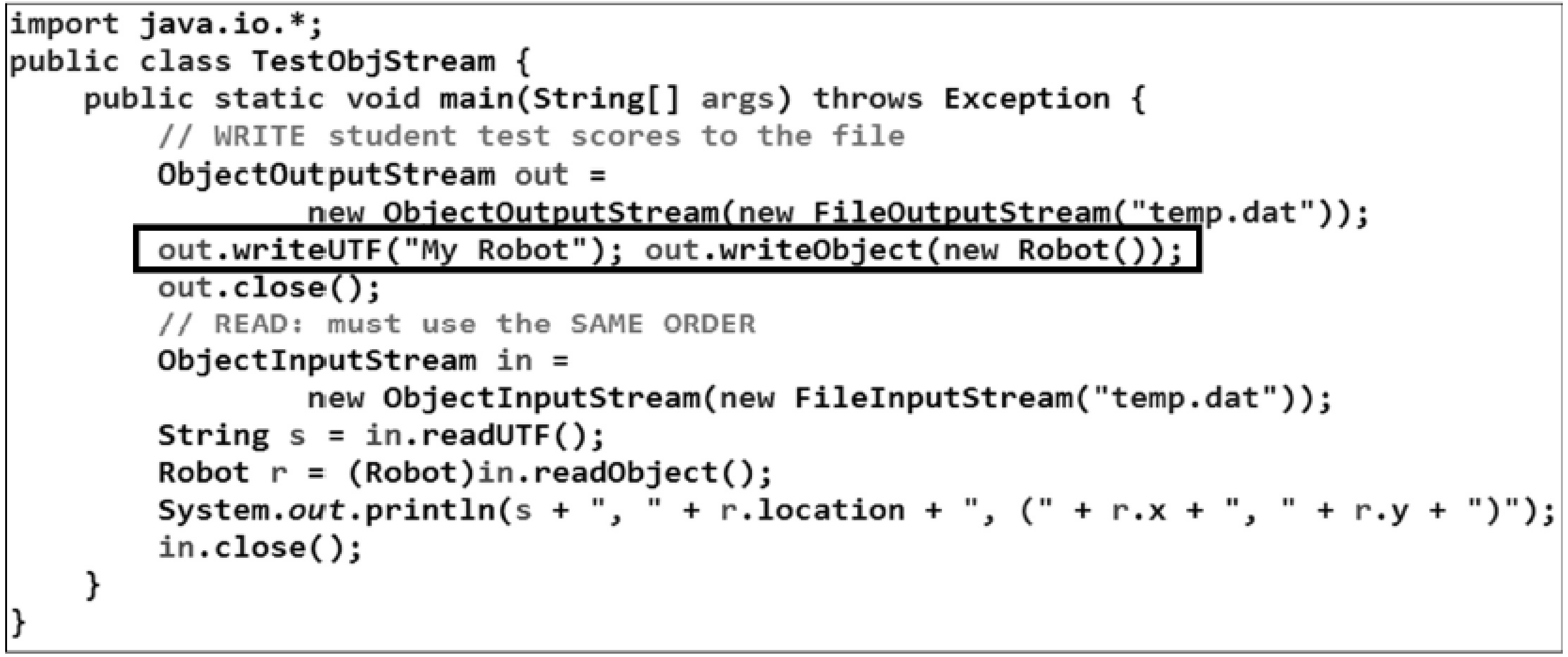
Analysis Questions: Binary I/O Example



```java
import java.io.Serializable;
public class Robot {
    int x = 10, y = 20;
    transient String location = "Canada";
    public void amethod(){}
}
```

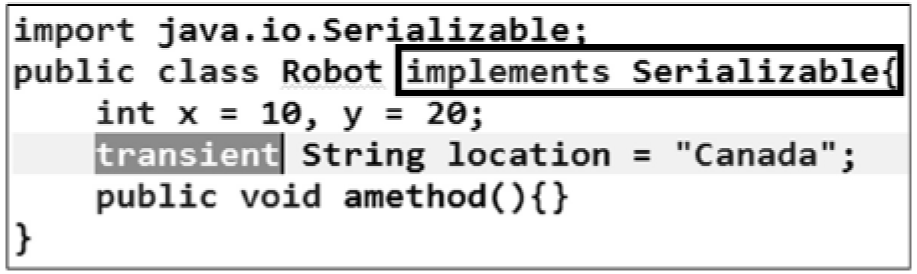
What is wrong ?

Can’t write Robot instance to file until it is Serializable



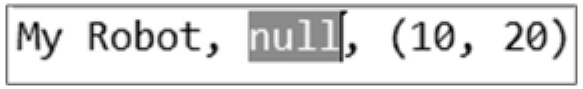
```java
import java.io.*;
public class TestObjStream {
    public static void main(String[] args) throws Exception {
        // WRITE student test scores to the file
        ObjectOutputStream out =
                new ObjectOutputStream(new FileOutputStream("temp.dat"));
        out.writeUTF("My Robot"); out.writeObject(new Robot());
        out.close();
        // READ: must use the SAME ORDER
        ObjectInputStream in =
                new ObjectInputStream(new FileInputStream("temp.dat"));
        String s = in.readUTF();
        Robot r = (Robot)in.readObject();
        System.out.println(s + ", " + r.location + ", (" + r.x + ", " + r.y + ")");
        in.close();
    }
}
```

Analysis Questions: Binary I/O Example

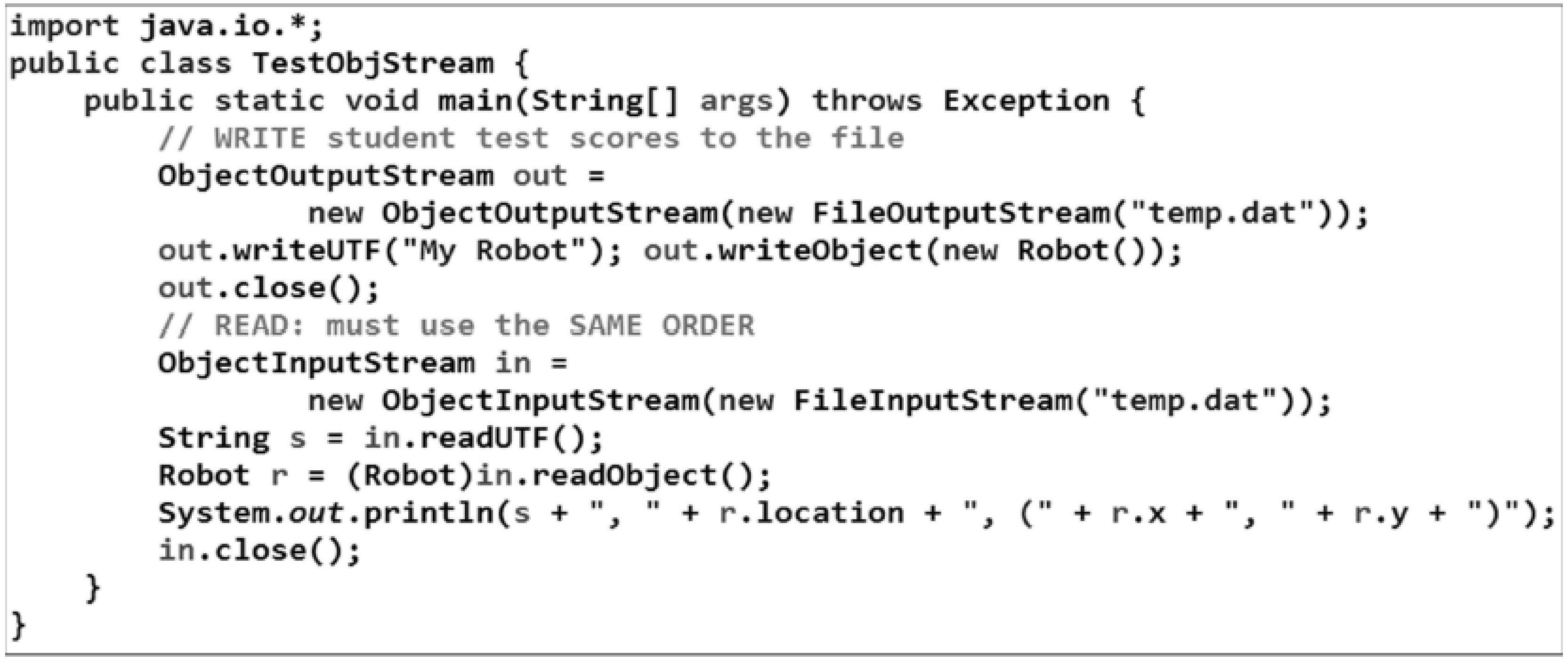


```
import java.io.Serializable;
public class Robot implements Serializable{
    int x = 10, y = 20;
    transient String location = "Canada";
    public void amethod(){}
}
```

What is the output?



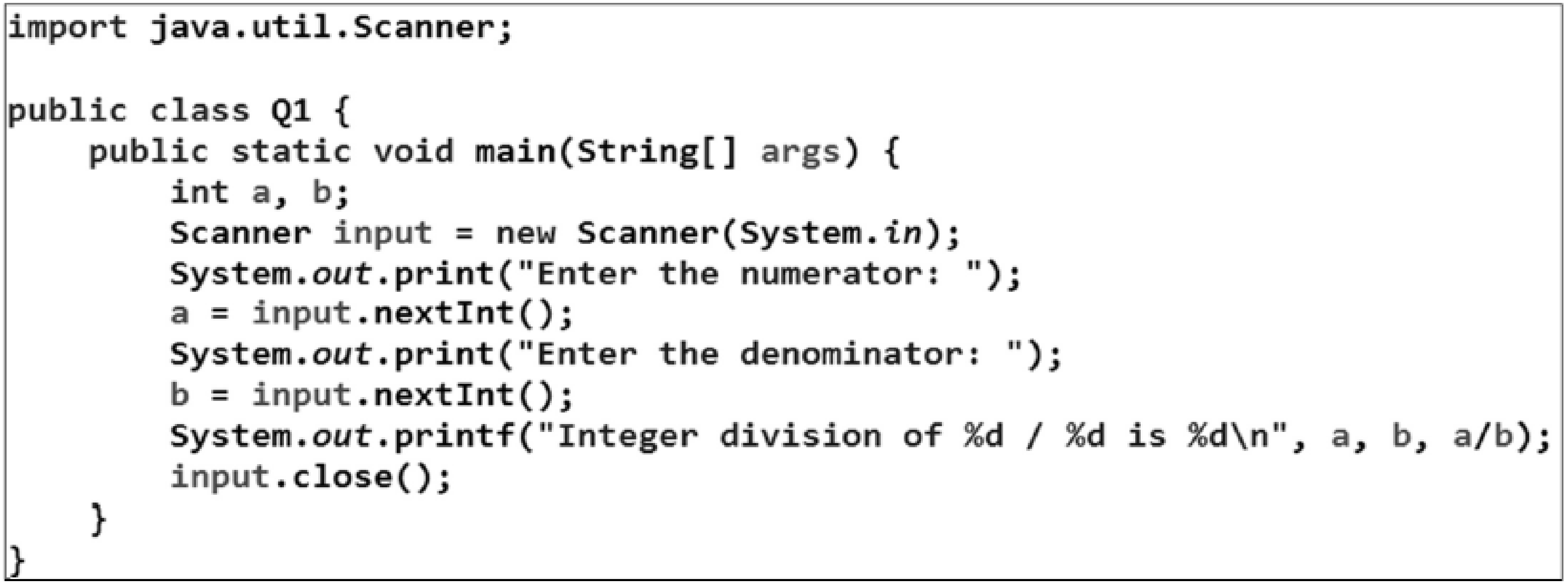
```
My Robot, null, (10, 20)
```



```
import java.io.*;
public class TestObjStream {
    public static void main(String[] args) throws Exception {
        // WRITE student test scores to the file
        ObjectOutputStream out =
                new ObjectOutputStream(new FileOutputStream("temp.dat"));
        out.writeUTF("My Robot"); out.writeObject(new Robot());
        out.close();
        // READ: must use the SAME ORDER
        ObjectInputStream in =
                new ObjectInputStream(new FileInputStream("temp.dat"));
        String s = in.readUTF();
        Robot r = (Robot)in.readObject();
        System.out.println(s + ", " + r.location + ", (" + r.x + ", " + r.y + ")");
        in.close();
    }
}
```

Coding Questions: Exception Handling Example

The following code assumes the user enters valid integer values. Modify the code so that the **code ensures** that the denominator is not equal to zero.

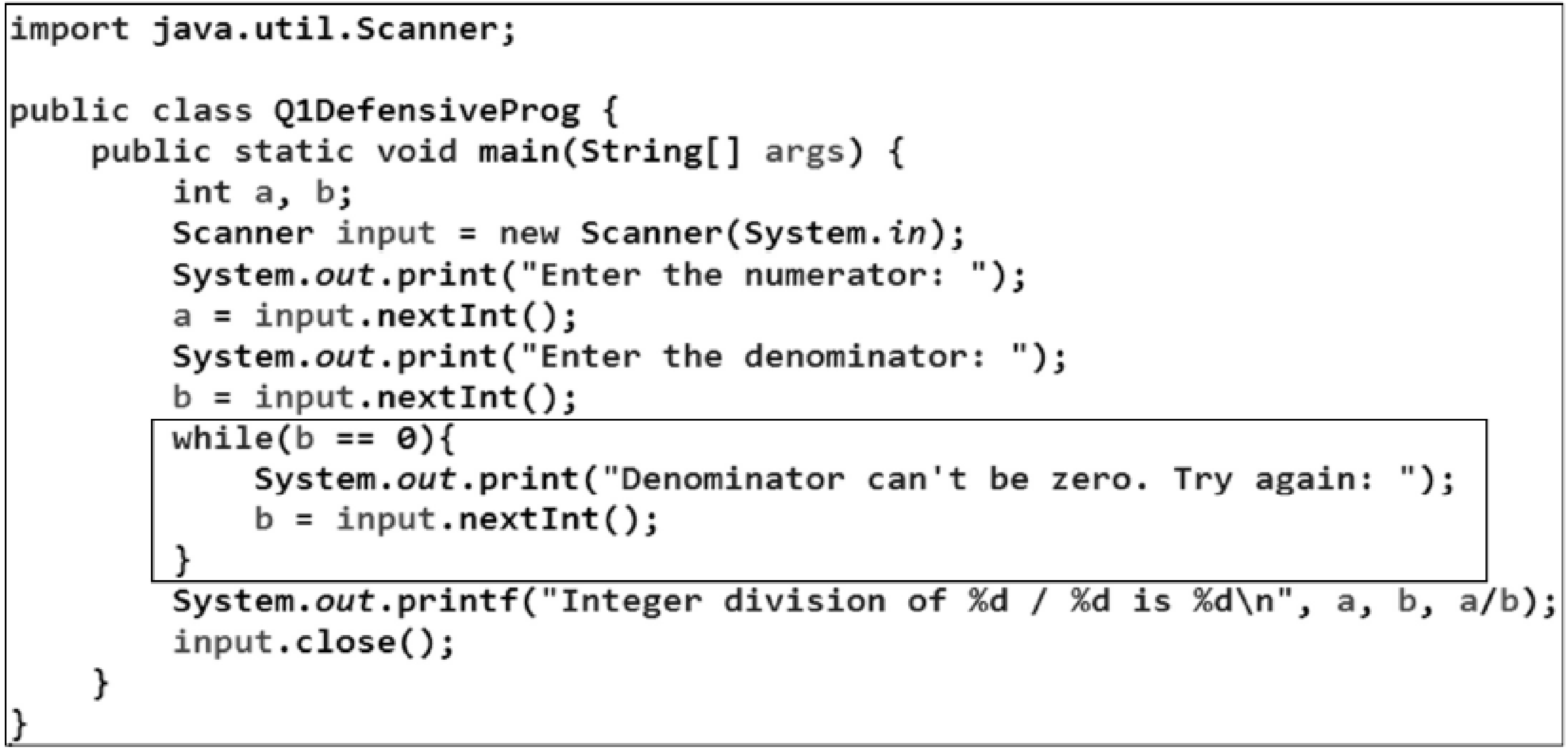


```java
import java.util.Scanner;

public class Q1 {
    public static void main(String[] args) {
        int a, b;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        a = input.nextInt();
        System.out.print("Enter the denominator: ");
        b = input.nextInt();
        System.out.printf("Integer division of %d / %d is %d\n", a, b, a/b);
        input.close();
    }
}
```

Solution

Solution 1: Defensive Programming

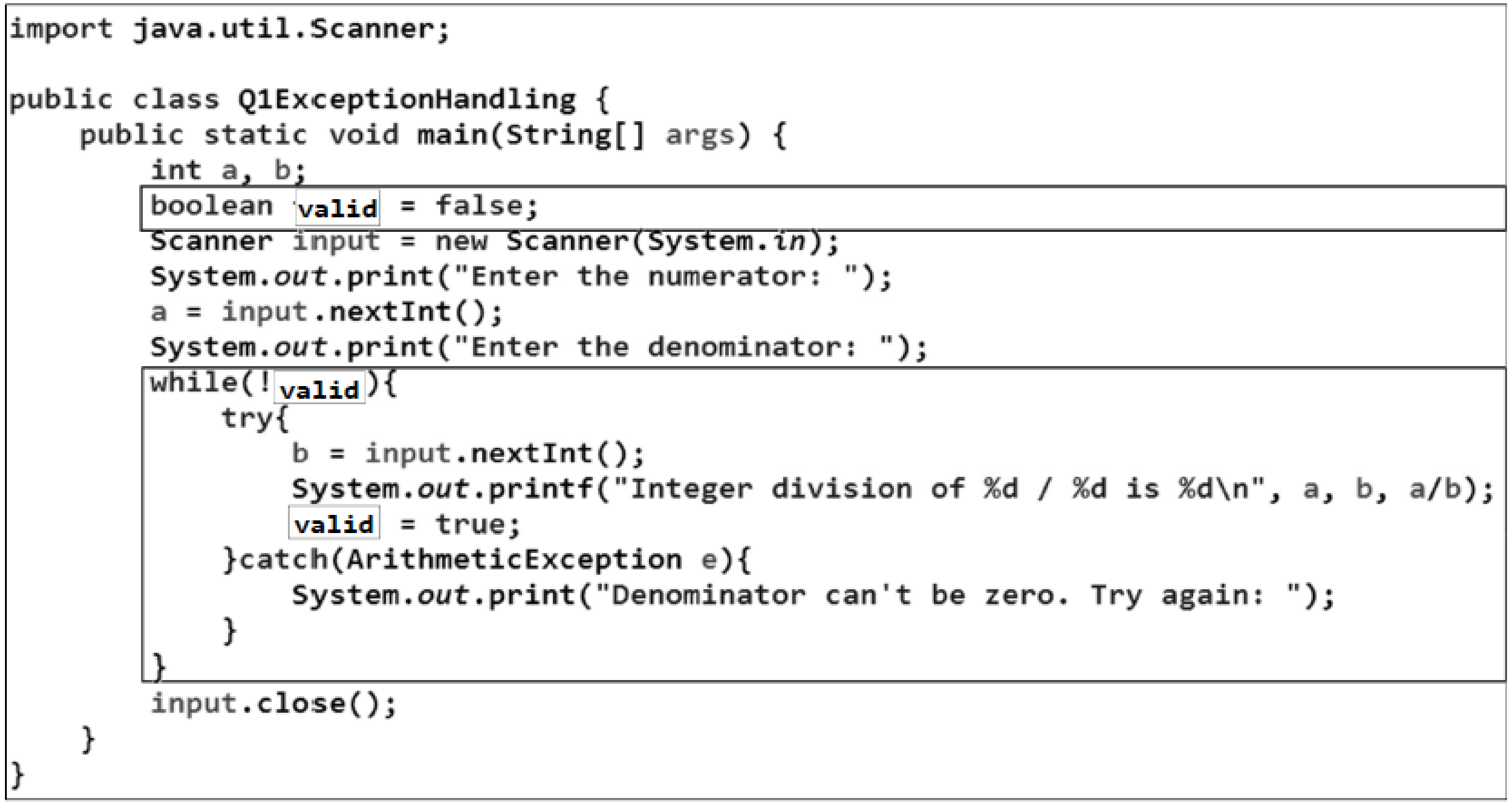


```java
import java.util.Scanner;

public class Q1DefensiveProg {
    public static void main(String[] args) {
        int a, b;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        a = input.nextInt();
        System.out.print("Enter the denominator: ");
        b = input.nextInt();
        while(b == 0){
            System.out.print("Denominator can't be zero. Try again: ");
            b = input.nextInt();
        }
        System.out.printf("Integer division of %d / %d is %d\n", a, b, a/b);
        input.close();
    }
}
```

Solution

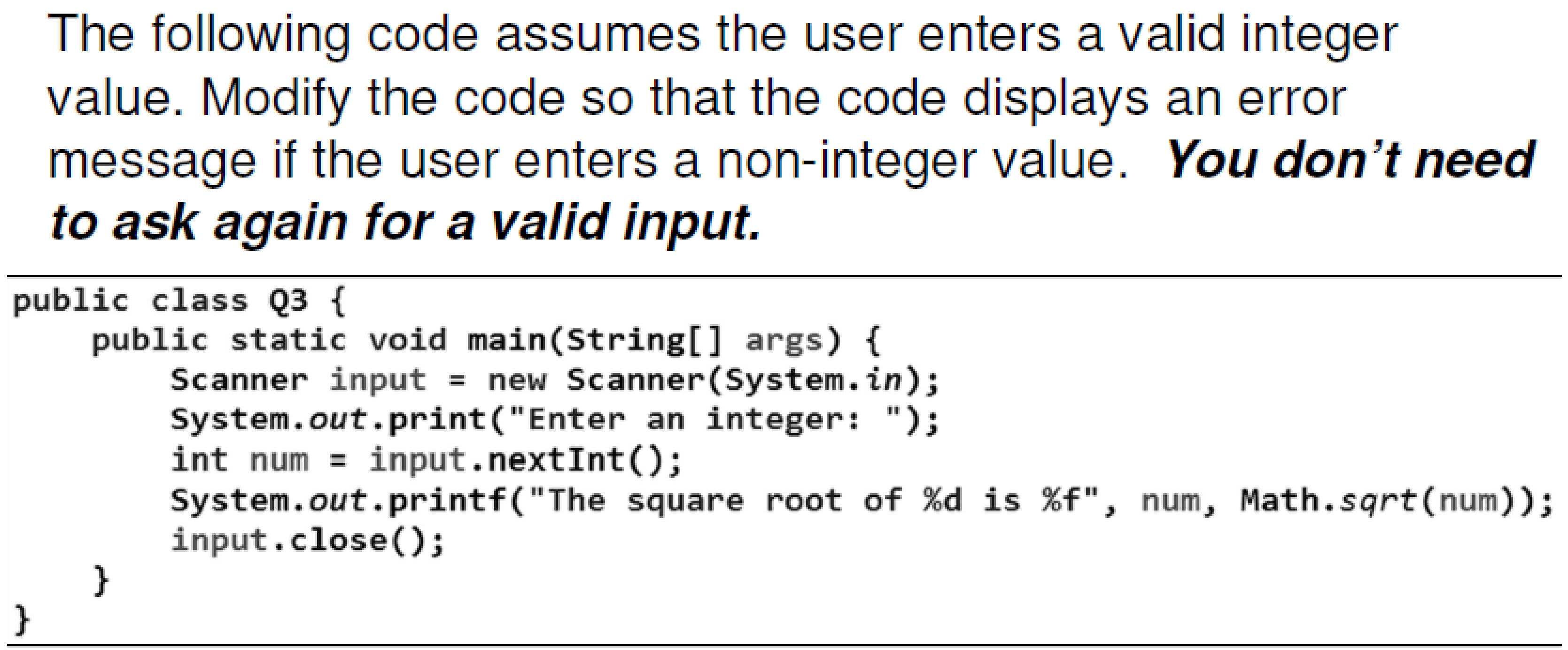
Solution 2: Exception Handling



```
import java.util.Scanner;

public class Q1ExceptionHandling {
    public static void main(String[] args) {
        int a, b;
        boolean valid = false;
        Scanner input = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        a = input.nextInt();
        System.out.print("Enter the denominator: ");
        while(!valid){
            try{
                b = input.nextInt();
                System.out.printf("Integer division of %d / %d is %d\n", a, b, a/b);
                valid = true;
            }catch(ArithmeticException e){
                System.out.print("Denominator can't be zero. Try again: ");
            }
        }
        input.close();
    }
}
```

Coding Questions: Exception Handling Example

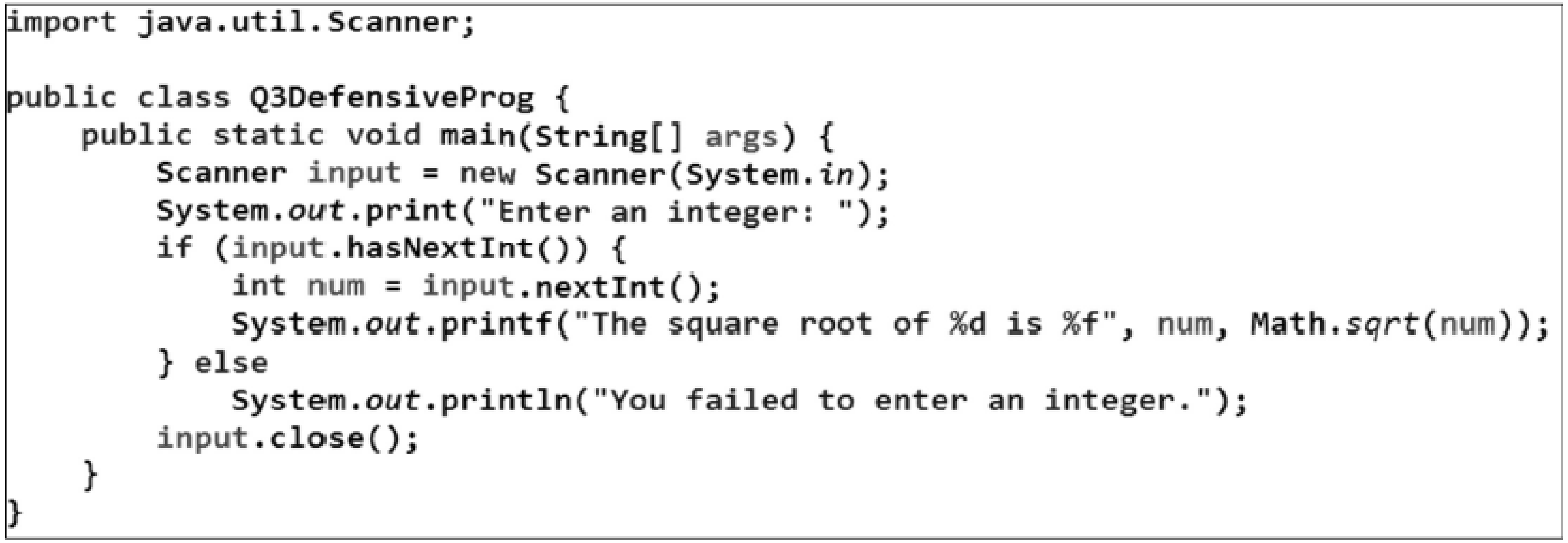


The following code assumes the user enters a valid integer value. Modify the code so that the code displays an error message if the user enters a non-integer value. ***You don't need to ask again for a valid input.***

```
public class Q3 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int num = input.nextInt();
        System.out.printf("The square root of %d is %f", num, Math.sqrt(num));
        input.close();
    }
}
```

Solution

Solution 1: Defensive Programming: check for an integer

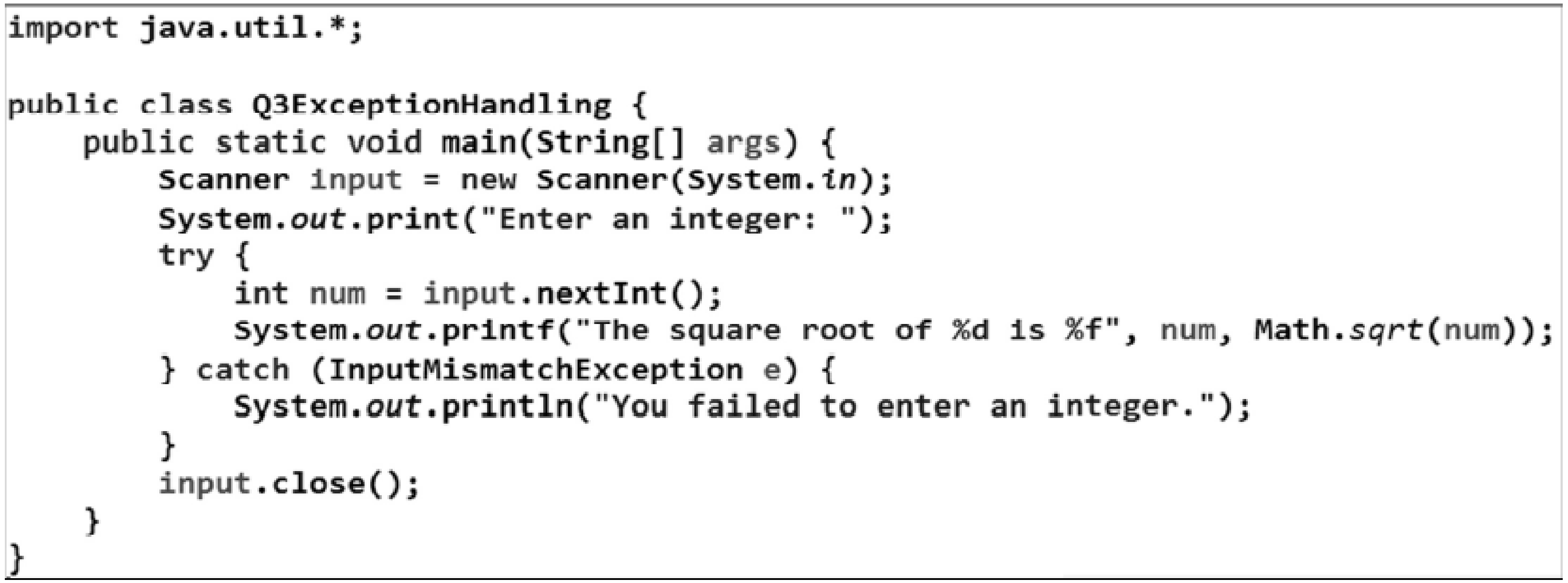


```
import java.util.Scanner;

public class Q3DefensiveProg {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        if (input.hasNextInt()) {
            int num = input.nextInt();
            System.out.printf("The square root of %d is %f", num, Math.sqrt(num));
        } else
            System.out.println("You failed to enter an integer.");
        input.close();
    }
}
```

Solution

Solution 2: Exception Handling: check for an integer



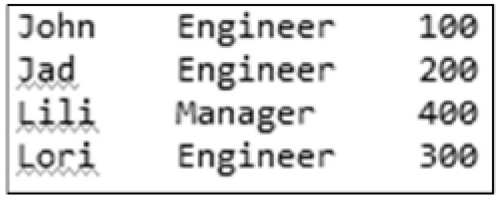
```java
import java.util.*;

public class Q3ExceptionHandling {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        try {
            int num = input.nextInt();
            System.out.printf("The square root of %d is %f", num, Math.sqrt(num));
        } catch (InputMismatchException e) {
            System.out.println("You failed to enter an integer.");
        }
        input.close();
    }
}
```

Coding Questions: Text I/O Example

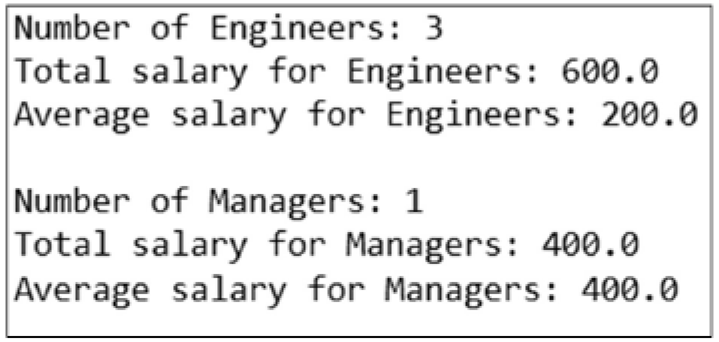
Assume a file *salary.txt* has the information of a company's employees. Each line in the file consists of a employee name, title, and salary. In this company, there are only two titles: Engineer and Manager. Write a program to display the number of Engineers and Managers as well as their total and average salaries.

salary.txt



```
John      Engineer    100
Jad       Engineer    200
Lili      Manager     400
Lori      Engineer    300
```

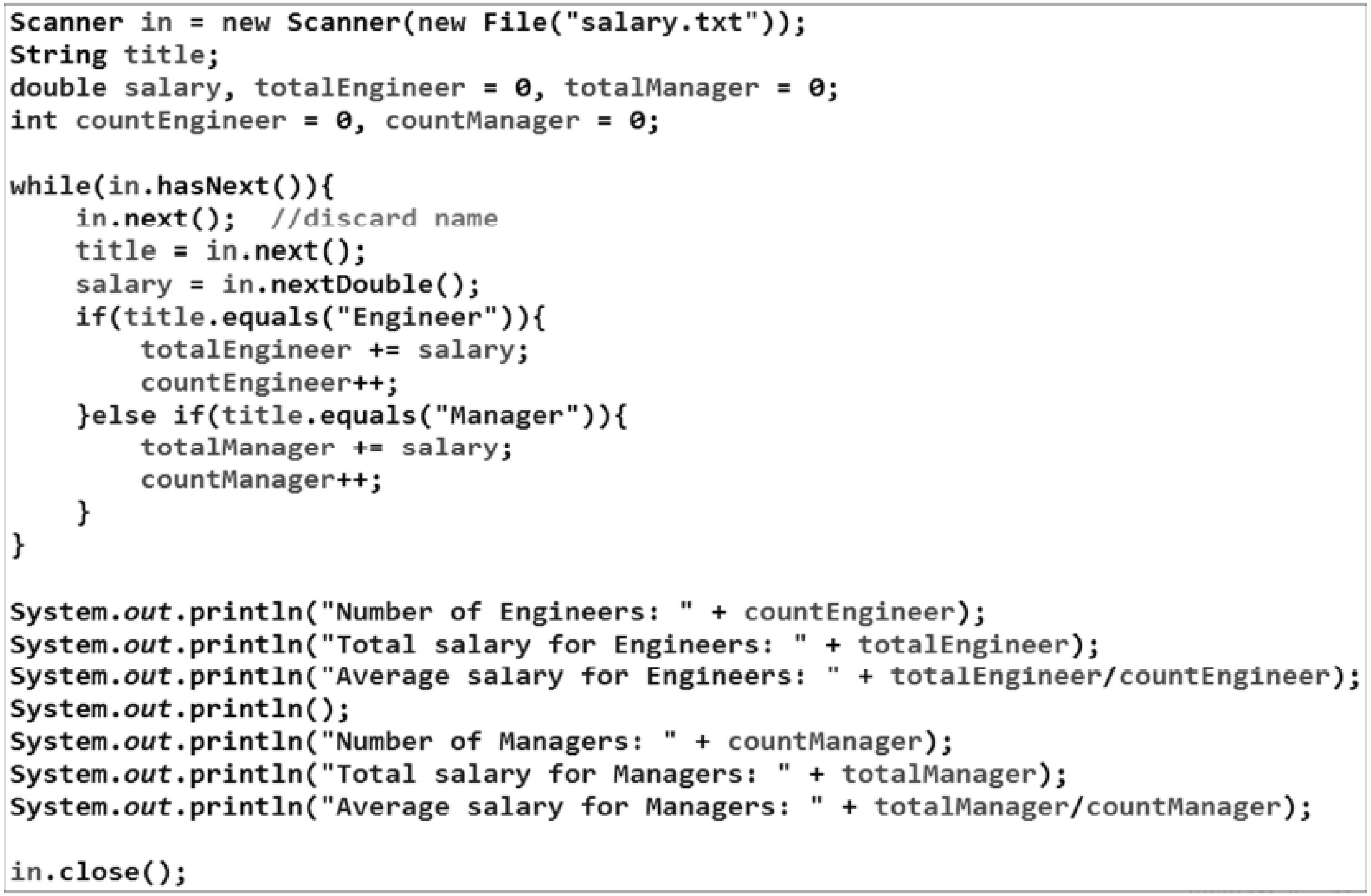
Output



```
Number of Engineers: 3
Total salary for Engineers: 600.0
Average salary for Engineers: 200.0

Number of Managers: 1
Total salary for Managers: 400.0
Average salary for Managers: 400.0
```

Solution



```java
Scanner in = new Scanner(new File("salary.txt"));
String title;
double salary, totalEngineer = 0, totalManager = 0;
int countEngineer = 0, countManager = 0;

while(in.hasNext()){
    in.next();  //discard name
    title = in.next();
    salary = in.nextDouble();
    if(title.equals("Engineer")){
        totalEngineer += salary;
        countEngineer++;
    }else if(title.equals("Manager")){
        totalManager += salary;
        countManager++;
    }
}

System.out.println("Number of Engineers: " + countEngineer);
System.out.println("Total salary for Engineers: " + totalEngineer);
System.out.println("Average salary for Engineers: " + totalEngineer/countEngineer);
System.out.println();
System.out.println("Number of Managers: " + countManager);
System.out.println("Total salary for Managers: " + totalManager);
System.out.println("Average salary for Managers: " + totalManager/countManager);

in.close();
```

Coding Questions: Binary I/O Example

Write a method that will split a given file into N pieces

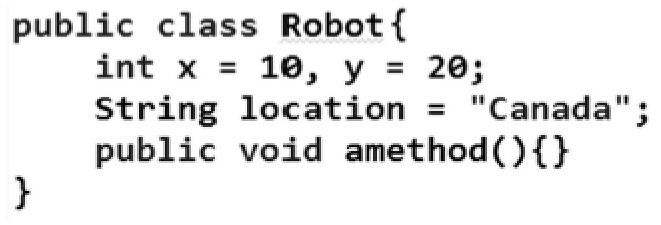
- Similar to the assignment question.

Write a method that reads the contents of a file and display them as:

- (a) integers
- (b) doubles
- (c) any given format (e.g. 1 integer, 2 doubles, 3 UTF characters)

Coding Questions: Binary I/O Example

Assume a Robot is defined by the following class



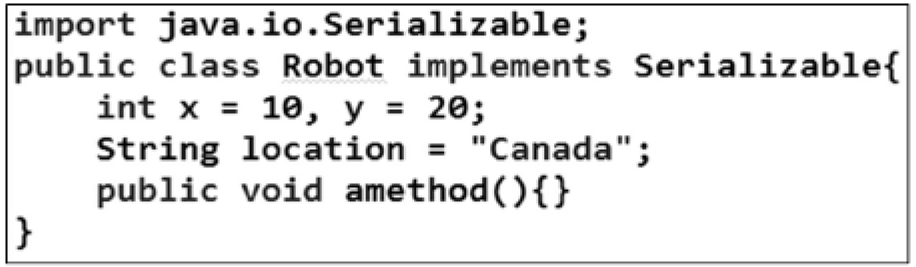
```
public class Robot{
    int x = 10, y = 20;
    String location = "Canada";
    public void amethod(){}
}
```

Write code to save to a file the following two items:

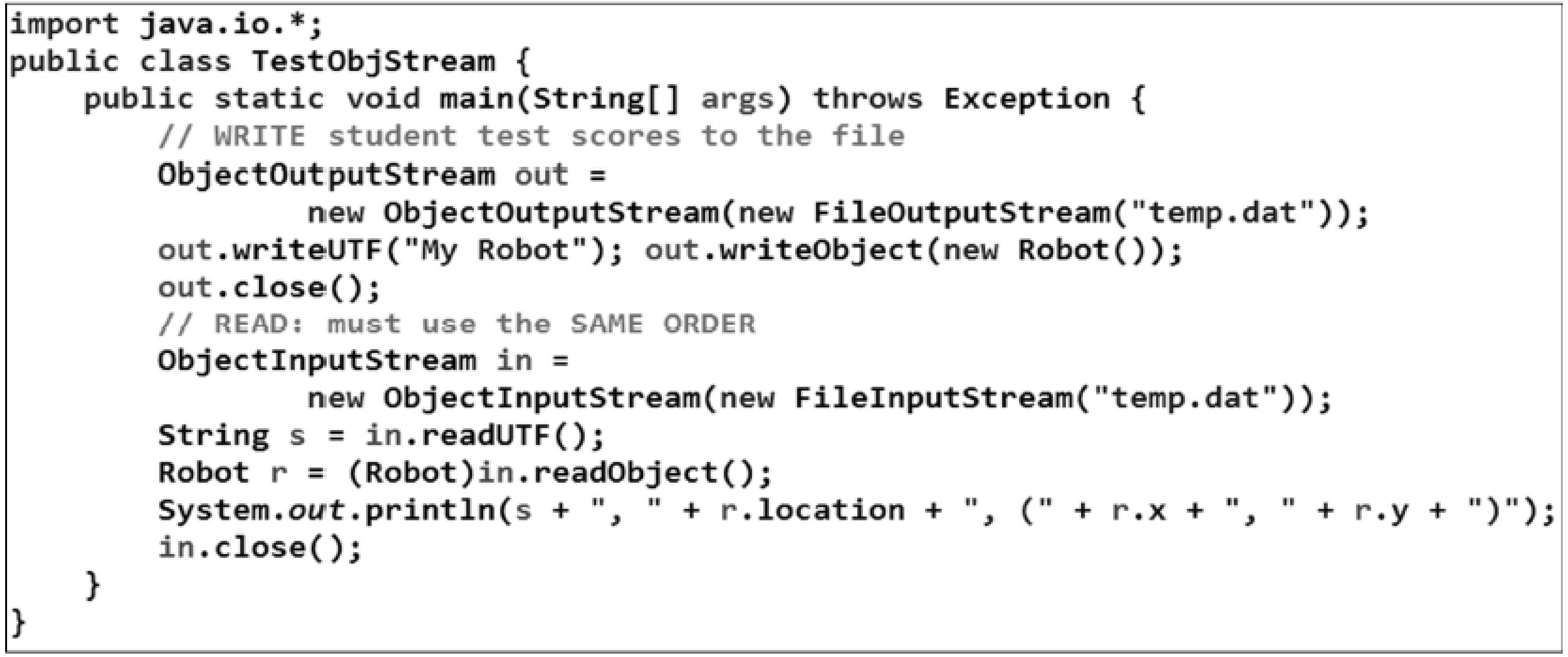
- A string, e.g. “My Robot”
- A Robot object (use default values for the attributes)

Then write code to load the string and the object and print their values on the console.

Solution

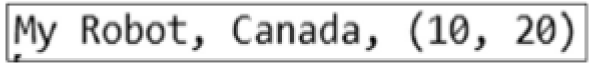


```
import java.io.Serializable;
public class Robot implements Serializable{
    int x = 10, y = 20;
    String location = "Canada";
    public void amethod(){}
}
```



```
import java.io.*;
public class TestObjStream {
    public static void main(String[] args) throws Exception {
        // WRITE student test scores to the file
        ObjectOutputStream out =
                new ObjectOutputStream(new FileOutputStream("temp.dat"));
        out.writeUTF("My Robot"); out.writeObject(new Robot());
        out.close();
        // READ: must use the SAME ORDER
        ObjectInputStream in =
                new ObjectInputStream(new FileInputStream("temp.dat"));
        String s = in.readUTF();
        Robot r = (Robot)in.readObject();
        System.out.println(s + ", " + r.location + ", (" + r.x + ", " + r.y + ")");
        in.close();
    }
}
```

Output



```
My Robot, Canada, (10, 20)
```