CSE2211: Object Oriented Programming and Design Methods

Quiz 1 (File I/O and Exception Handling) Date: 23.09.2018

1. Suppose the input contains the characters 995.0 Fred. What are the values of number and input after this code fragment?

int number = 0;

if (in.hasNextInt()) {

number = in.nextInt();

}

String input = in.next();

**Answer**: Because 995.0 is not an integer, the call in.hasNextInt() returns false, and the call in.nextInt() is skipped. The value of number stays 0, and input is set to the string "995.0".

1. Suppose the input contains the characters 6E6 $6,995.00. What are the values of x1 and x2 after this code fragment?

double x1 = in.nextDouble();

double x2 = in.nextDouble();

Answer:

x1 is set to 6000000. Because a dollar sign is not considered a part of a floating-point number in Java, the second call to nextDouble causes an input mismatch exception and x2 is not set.

1. Suppose balance is 100 and amount is 200. What is the value of balance after these statements?

if (amount > balance){

throw new IllegalArgumentException("Amount exceeds");

}

balance = balance - amount;

Answer: It is still 100. The last statement was not executed because the exception was thrown.

1. Consider the method

public static void main(String[] args) {

try {

Scanner in = new Scanner(new File("input.txt"));

int value = in.nextInt();

System.out.println(value);

} catch (IOException exception) {

System.out.println("Error opening file.");

}

}

Suppose the file with the given file name exists and has no contents. Trace the flow of execution.

**Answer**: The Scanner constructor succeeds because the file exists. The nextInt method throws a NoSuchElementException. This is *not* an IOException. Therefore, the error is not caught. Because there is no other handler, an error message is printed and the program terminates.

1. What is wrong with the following code, and how can you fix it?

public static void writeAll(String[] lines, String filename) {

PrintWriter out = new PrintWriter(filename);

for (String line : lines) {

out.println(line.toUpperCase());

}

out.close();

}

Answer: There are two mistakes. The PrintWriter constructor can throw a FileNotFoundException. You should supply a throws clause. And if one of the array elements is null, a NullPointerException is thrown. In that case, the out.close() statement is never executed. You should use a try-with- resources statement.

1. What happens if you try to open a file for writing, but the file or device is write- protected (sometimes called read-only)? Try it out with a short test program.

**Answer:** If you open a file for reading and the file doesn't exist, then an exception is thrown. If you open a file for writing and the file doesn't exist, then the file is created with 0 length. The system throws an IOException in each situation

1. If a program Woozle is started with the command

java Woozle -Dname=piglet -I\eeyore -v heff.txt a.txt lump.txt

what are the values of args[0], args[1], and so on?

Answer:

args[0]: -Dname=piglet

args[1]: -Ieeyore

args[2]: -v

args[3]: heff.txt

args[4]: a.txt

args[5]: lump.txt

1. What is the purpose of the try-with-resources statement? Give an example of how it can be used.
2. The following code tries to close the writer without using a try-with-resources statement:

PrintWriter out = new PrintWriter(filename);

try {

//Write output.

out.close();

} catch (IOException exception) {

out.close();

}

What is the disadvantage of this approach?

**Today’s Lab work**

