

CSE 11
Winter 2017
Programming Assignment #1

START EARLY!

100 Pts

Due: 25 JAN 2017 at 11:59pm (2359)

Covers Chapters: ZY 1-3

This is a combination of a programming assignment and ungraded exercises

Exercises are optional but recommended. They are not graded.

Exercise #1: Make certain you can log into the campus lab computers in the CSE building. All of your programming should be done in Linux. ALL PROGRAMS MUST BE TURNED IN FROM LAB MACHINES. You may be remotely logged into the machines when you turn in.

Exercise #2: If you have a Windows or Mac laptop and would like to have Linux environment for development, please go to the class website and look at the page called “Run a Development VM on your Laptop”

Exercise #3: Get some basic familiarity with one of `emacs` or `vim`. One easy way to do so is to run through the tutorial called “`vimtutor`” in the lab to learn more about how to use `vim`. Also, running the `gvim` command will provide a graphical version of `vim`. For `emacs`, running the `emacs` command will also open a more familiar graphical editor.

Exercise #4:

Create a subdirectory in your home area called `PR1`

```
$ cd $HOME
$ mkdir PR1
$ cd $HOME
```

Download `HelloWorld.java` and `Twist.java` from the class web site and place them into your `PR1` subdirectory.

Following the directions in the comments of each program, compile and run these programs

PROGRAMMING Assignment (100 pts) : FourDice

You are to create a program called `FourDice.java`. You will use a single random number generator instance from `java.util.Random` to generate four different random numbers. The program first reads (from `System.in` which is also called “standard input” or often `stdin`) a single integer. This input might be used as a numeric seed for a single random number generator instance (see below).

- Your program uses the generator to create four different random integers in the range `[1,6]` (closed range, includes the endpoints 1 and 6).
- It prints out the four numbers generated (the numbers are like rolls of dice).
- It finds the Minimum and Maximum numbers generated, and prints them out

- If all four numbers are identical, it prints out “Go to Vegas!”
- If exactly three of the four numbers are identical, it prints out “Three of a kind.”
- If it is neither of the above, it prints out “Not so lucky.”

See the example output for the precise format that your program should generate. Your program MUST follow this format for the auto-grader to work properly. Spaces are Important, as is spelling and punctuation.

Also, see the Requirements section for additional essential detail.

Here are a few runs of the program. **Boldface** indicates input by the user. [PR1] \$ is the command-line prompt. Notice that running with seed == -10 gives different answers each time the code is run. That is correct.

```
[PR1]$ java FourDice
10
Seed : 10
Roll 1 : 4
Roll 2 : 1
Roll 3 : 4
Roll 4 : 1
Smallest Roll : 1
Largest Roll : 4
Not so lucky.
```

```
[PR1]$ java FourDice
13
Seed : 13
Roll 1 : 5
Roll 2 : 5
Roll 3 : 6
Roll 4 : 5
Smallest Roll : 5
Largest Roll : 6
Three of a kind.
```

```
[PR1]$ java FourDice
55
Seed : 55
Roll 1 : 2
Roll 2 : 2
Roll 3 : 2
Roll 4 : 2
Smallest Roll : 2
Largest Roll : 2
Go to Vegas!
[PR1]$
```

```
[PR1]$ java FourDice
```

```

-10
Seed : -10
Roll 1 : 4
Roll 2 : 6
Roll 3 : 5
Roll 4 : 3
Smallest Roll : 3
Largest Roll : 6
Not so lucky.

[PR1]$ java FourDice
-10
Seed : -10
Roll 1 : 1
Roll 2 : 1
Roll 3 : 1
Roll 4 : 1
Smallest Roll : 1
Largest Roll : 1
Go to Vegas!

```

Key requirements of your program

- 1) There is single integer read from standard input (`System.in`), call this the *seed*.
- 2) You generate exactly one instance of the `java.util.Random` class
 - a) If the *seed* is greater than 0, use the *seed* as the seed of your `java.util.Random` instance.
 - b) If the *seed* is less than or equal to 0, then you should not use the *seed* as the seed of your `java.util.Random` instance. You should not use any explicit seed in this case.
- 3) Print out the seed, generated numbers, and output as above. Your program is partially autograded and following the format rules are essential for the autograder to work properly.
 - a) Include a single space on either side of the “:” (colon) on each line of output.
 - b) Include no extra output.
 - c) Use `System.out.println` for all lines of output
- 4) Compute the smallest and largest rolls and print out exactly as above
 - a) Feel free to use the `Math.min` and `Math.max` methods provided by `java.lang.Math`
- 5) If all dice rolls were identical, print the phrase “Go to Vegas!”
- 6) If *exactly* three of the four dice rolls are identical print the phrase “Three of a kind.”
- 7) If neither of the above are true, print the phrase “Not so lucky.”
- 8) Random numbers (rolls) are in the range [1,6]. (closed interval, 1 and 6 are possible numbers)
- 9) The class must be called `FourDice` and must be in the file `FourDice.java`
- 10) **Do not add any extra lines** of output. Your output should look just like the examples above.

- 11) You must use constants in your program (variables declared as `final`). Don't use so-called "magic numbers" in your code. A magic number is a hard-coded value that may change at a later stage, and can therefore be more difficult to update. Instead define constants and use that constant in your program. Define constants maximum allowable random number. 0,1, & 2 are generally not considered "magic" numbers in programs
- 12) You do NOT have to check if the input is actually an integer. We will test your program only with valid integer inputs that do not overflow Java's `int` datatype.

Formatting/Commenting Requirements of your Program

- 1) Your code must be properly indented. You may use spaces or tabs. Do NOT mix spaces and tabs for indentation. Choose one or the other and be consistent. (10 Points).
- 2) You must put in comments of your program at the top of the file (10 Points)
Name : <Your Full Name>
Email : <Your UCSD Email>
ID : <Your Student ID, e.g. A123456789>

Hints:

- 1) A `return` statement can be used to stop execution of your main method and prevent it from executing any further statements. This can be helpful in debugging
- 2) Build your program in stages. You don't have to write the entire program before you begin to debug. Write part of the program, test it, then add another part, test it, and so on.
- 3) Generate only one instance of the `Random` class and use that instance to generate all four of the required random integers. If you create and use more than one instance of the `Random` class, then statistics for your program when run multiple times will be incorrect.
- 4) Think a bit about how to determine if exactly three of four dice are identical. You have computed the minimum and maximum values rolled. Once you have computed these values, have your program count the number of rolls that are equal to this minimum value AND the number of rolls that are equal to the maximum value. What must be true if you have four of a kind? What must be true if you have exactly three-of-a-kind?

Turning in your Program

YOU MUST BE LOGGED INTO THE LAB MACHINES FOR THIS TO WORK. PLEASE VERIFY WELL BEFORE THE DEADLINE THAT YOU CAN TURNIN FILES

You will be using the "bundlePR1" program that will turn in the following file

FourDice.java

No other files will be turned in and they **must be named exactly as above**. BundlePR1 uses the department's standard turnin program underneath.

To turn-in your program, you must be in the directory that has your source code and then execute the following

```
$ bundlePR1
```

Sample output of a successful turnin

```
$ bundlePR1
Good; all required files are present:

FourDice.java

Do you want to go ahead and turnin these files? [y/n]y
OK. Proceeding.

/home/linux/ieng6/cs11wa/public/bin/checkCompilePR1.py
/tmp/b93C0fACt/cs11wa.tar
Unpacking files ... tar xfbp /tmp/b93C0fACt/cs11wa.tar --
dir=/tmp/tmpxp_e8C
Copying support files
Checking Compilation of FourDice.java ...
Compilation of FourDice.java OK
...SUCCESS

Performing turnin of approx. 3481 bytes (+/- 10%)
Copying to /home/linux/ieng6/cs11wa/turnin.dest/cs11wa.PR1
..
Done.
Total bytes written: 3072
Please check to be sure that's reasonable.
Turnin successful.
```

Sample output of an unsuccessful turnin (Compiler error)

```
$ bundlePR1
Good; all required files are present:

FourDice.java

Do you want to go ahead and turnin these files? [y/n]y
OK. Proceeding.

/home/linux/ieng6/cs11wa/public/bin/checkCompilePR1.py
/tmp/3ASK_x6q0J/cs11s.tar
Unpacking files ... tar xfbp /tmp/3ASK_x6q0J/cs11wa.tar --
dir=/tmp/tmpUmebPL
```

```

Copying support files
Checking Compilation of FourDice.java ...
Compilation of FourDice.java failed
Failed Compilation Test
=== Errors seen while Compiling ===
FourDice.java:9: error: ';' expected
import java.util.Scanner
                        ^
1 error
=== End of Errors seen while Compiling ===
COMPILER ERRORS, you CANNOT turn in

```

You can turn in your program multiple times. The turnin program will ask you if you want to overwrite a previously turned-in project.

ONLY THE LAST TURNIN IS USED!

Suggestion: if you have classes that compile and do some or most of what is specified, turn them in early. When you complete all the other aspects of the assignment, you can turn in newer (better) versions.

Don't forget to turn in your best version of the assignment.

Frequently asked questions

- 1) **Does my program have to achieve the identical results of the sample output when the seeds are the same?** *No. You may be using your Random number generator slightly differently than the code used in the example output. **Your code should always give the same output when using the same (positive) seed.***
- 2) **How important are the colons and fields in the comments?** *Critical. You will lose points if you don't have a : separating your Name from your full name, divisors and numbers. etc.*
- 3) **What if my program doesn't compile, can I get partial credit?** *No.*
- 4) **I program in an IDE like Dr. Java, do I have to use the command line?** *Yes.*
- 5) **Do I need to check for bad input (like typing in a string of letters instead of an integer)?** *No.*
- 6) **How will this be graded?** *20 points of your 100 points are identified above. We will check the form of your program and look for correct functionality. For example it should generate random numbers. We'll check for randomness by running your code hundreds of times. If we run your program 1000 times, then Four-of-a-kind should show up about 8 times and Three-of-a-kind should show up about 100 times. We'll also check that you get the right answer each time your code runs. The readers will verify that you didn't use magic numbers and declared constant(s) where appropriate*
- 7) **Can I get partial credit?** *If your program compiles, yes. If your program does not compile, you cannot turn it in.*
- 8) **Do I get graded on comments?** *Only the required Name, email, ID will be checked for this assignment. The readers will give feedback on the style of your program to help you with future programs. Try putting a few comments in your program to describe, in English, what the next several lines of code are doing. For example, you will have a sequence of statements that need to be*

executed to compute the min and max values. Something like the following in your code is about the right amount of commenting.

```
// find min and max values rolled
```

```
... java statements that compute the min and max of your rolls ...
```

9) **Do I get graded in proper indentation? Yes!**

10) **If I know about for loops and/or arrays can I use them in my solution? Yes.**

11) **If I don't know about for loops and/or arrays, do I need to learn about them first? No.** *The professor's solution uses neither for loops nor arrays. We'll cover these concepts later in the course. The assignment is designed to be completed without these language features.*

12) **How long is this program? The professor's solution including comments and blank lines is 79 lines.**

13) **Can you show me a cool unix command line "trick" to run my program hundreds of times with a different seed each time? Yes. Try this:**

```
for i in `seq 1 100`; do echo $i | java FourDice; done)
```

14. **Do I need to "understand" that command line right now? No.** *However, in English it says "generate a sequence of numbers 1 to 100, use each of those numbers in turn as the input to the FourDice program."*

15. **How do I run my program the first time? I get an error like `Error: Could not find or load main class FourDice.java`.** Make sure to compile the program before you run it the first time, *and* after you make each change you want to test. Compile with this command at the command line:

```
javac FourDice.java
```

START EARLY! ASK QUESTIONS!