Class 4/5 – Exceptions, I/O

Exercise 1:

From the RPG game we created in the previous class:
1. Create first a simple Jar file (making sure to add sources as well),
2. Then create an executable Jar file with the same name.
Double click each one to see what happens, why do they not run? Try through a console next. Also check their contents. What are the differences between the two?

Exercise 2:

Write a program for dividing two integers, that you pass as an argument at run time (using Eclipse configurations, or by running through a console). Add exception handling for improper input, and give appropriate information to the user (can also print the call trace).

Exercise 3:

We will now write our first code for reading keyboard input. Create a new package called session5.ioExamples. Inside create a class called SimpleKeyboardInput.

1. In the main function, write code that prompts the user to type something. Then read that and output what the person wrote. Use System.out.print instead of System.out.println if you want your prompt to be at the same line as the user input.

2. Now ask the user to input a specific word, e.g. *Hello*. If they type something different, continuously ask them to type until they get it right!

3. Be a bit more forgiving with your user, by allowing them to have the correct word, even if they get the case (upper vs lower) wrong. I.e., accept as correct Hello, hello, hellO, etc.

Exercise 4:

We will now learn how to read specific input type (e.g., int, double, boolean). Create a new class called SimpleTypeKeyboardInput.

1. In the main function, write code that prompts the user to type a specific type, an integer. Then read that, and convert it to that type.

2. Add an exception to give the user a more informative message when they type inappropriate input.

3. Adjust your program so that you ask your user to write several integers (as many as they wish). Read their input, convert it to integers, and store them in an array.

Exercise 5:

We will now upgrade yet again our RPG. If you want to keep previous versions of your code, create a new package called class5.ioRPG and create for now a new class called RPGio. Instead of hardcoding the characters of our game, we will ask the user to input them at the command line.

Part one
In your RPG class write code that asks the user to input the parameters of a character in turn (e.g., first_name, profession, health, mana). Explain in the text you use to prompt your user what is the format you are expecting (try something like the example above). For now just repeat back to the user what their input is.

Part two
1. Extend your code, by trying to convert the input to the appropriate format (e.g., mana to integer), and check that the professions are correct. Use the appropriate conversion functions and string manipulation functions to help you check, or throw appropriate exceptions.

Part three
If you are sure your user input is correct, create the character and add them to your RPG!

Part four
Extend your input, such that the user can create new characters for 4 times until. Run the RPG with all the new characters. Consider organizing your code better, e.g., moving the code that converts a line into a character in a different method.

Part five
Extend your input, so that users can continuously input characters until they input a special sequence (e.g., when they write "done"). Ooof, too many things to write !!!

Exercise 6:

Create a file named "readme.tx" and put it in your project directory. Read it and print its contents.

Exercise 7:

Create an array of strings and print it into a file.

Exercise 8:

We will upgrade a bit more our RPG, by reading characters from a file (instead of hardcoding them, or of asking the user for direct input).

1. Create a new txt file by hand with your characters (one character per line), read them, create their respective object in your program and then ran your RPG. Make sure to check properly their profession.

Note, for each line representing a character, you can separate attributes by ",". This creates a csv (comma separated values) text file that you can save as "characters.csv"

2. While your program is running, keep a record of the elimination of players, and of the winner. Do this by writing this information into a new file on the disk.