Exam PGR112 – Object orientated programming V2024

Candidate number: 87

Exceptions:

An exception is something a programming language will throw if there is an error. This way you might get a hint on what is wrong with the code. I came across a few exceptions that java threw in my code, and I decided on handling them a few different ways. My "Menu.java" class was prone to errors as the user will be prompted to type into the terminal, and I wanted to prevent the program from crashing or misbehaving if the user input did not match what they were supposed to enter. In my switch statement, I have set the user input to "toLowerCase", this way the user is free to use upper or lower case, without breaking the program.

In the same method at case "b", the user is prompted to enter a number of what year they want to look at. Here if any letters are typed in, the user would get a message saying they must enter a valid number and then the menu will show up again. Previously the terminal would crash because of the "NumberFormatException", but now the exception is being handled, and it is more user friendly.

At the end of the method I used "default" that is an edge-case handling in switch statements. Here I can print out to the user that they must use a valid menu option and then show them the menu again.

In part one of this task there were several exceptions that could happen around database queries, connections and file reading. The program crached on me a few times, but i ended up providing better error messages than what Java would give. This can be seen in the "Person" class, under the method "checkIfPersonIdExists()". This would help me greatly during the making of the program but could also help others who might use or look at my code later. This error handling can be seen in all my methods.

Encapsulation:

Encapsulation means to separate the code into smaller "capsules", make class variables private, use getters and setter and be very selective of what methods you want to be static. This way you can protect your data in the program. I have done this by separating my code into several classes. Each class handles very specific code and methods that is meant just for them. My class variables are all private and methods that use static only use getters to collect my data.

I have not used any setters in my code this time, because I had no use case for them, as my constructors did what I needed.

Inheritance:

Inheritance means that classes can inherit attributes and methods from another class. I set my “Find” class to be the “superclass” or a parent in other words. I have three other classes that inherit from this superclass. Those classes are “Coin.java”, “Weapon.java” and “Jewelry.java”. All these three classes were supposed to have many of the same attributes and that is why I made the superclass “Find”. Now the three subclasses could inherit some attributes, but still have some different attributes what are unique to them.