This is a review project for the Java Assessment taken near the end of the Java II Module. This review covers:

Primitives

Objects

Console IO

Controls Statements (If, else, if-else, switch)

Loops (while, do-while, for)

Strings

Methods

Arrays

OOP/OOD (abstraction, encapsulation, inheritance, and polymorphism)

<https://howtodoinjava.com/oops/object-oriented-principles/>

Interfaces and Abstract Classes

Collections

Annotations

Error Handling

File IO

Clone the master branch of the europa-java-exercises repository into a new directory. The repository is at <https://github.com/CodeupClassroom/>. After cloning the repository, create a folder in the src directory named “review”

Exercise 01: The IntClass class.

**This exercise will test Primitives, Objects, Methods, Arrays, Loops**

For this exercise we are going to create a wrapper class for the primitive type “int”. The class will have a getter and a setter and other methods for doing operations common to integers.

Step 1) Create a class name IntClass in the review folder.

Step 2) Add an attribute named “value” of type int.

Step 3) Add getters and setters for the attribute.

Step 4) Add an instance method named calcSum, to return the sum this instance with the sum of another instance passed as a parameter.

public int calcSum(IntClass myInt);

Step 5) Create a class method named calcSum to return the sum of two instances of the IntClass.

public static int calcSum(IntClass valOne, IntClass valTwo);

Step 6) Create a constructor to create a new instance of IntClass based on the value of a passed down int.  public IntClass(int val);

Step 7) Create a class method to take in an array of int and return a new instance of IntClass with the value equal to the sum of the array.

public static IntClass sumArray(int[] x);

Step 8) Create a main method to test each of these methods.

Exercise 02: The Calc class.

**This exercise will test Console IO, Control Statements, Loops, Arrays, and Error Handling.**

Create class to implement a calculator. Implement the add, subtract, multiply, and divide methods. Put in controls for errors like dividing by zero. All of the operations should use integers.

Create a main method to test the class interactively. Allow the user type “exit” to exit the application, or one of the symbols , “+”, “-”, “\*”, and “/”,  to select any of the four operations. Use the symbols in a switch statement to decide which operation to use. Once the user selects the operation, allow the user to enter two numbers. Include error handling using a try catch in case the user enters something other than a number. Use the Calc class method  to evaluate the operation, and print the result. After printing the result return to the menu.

Exercise 03: The RealCalc class.

**This exercise will test Inheritance.**

Create a second class named RealCalc which extends the Calc class keeping all of the int methods, and overloads these methods with methods that support real numbers.

Create a main method to test the new class interactively. Add 4 more options for the four new methods.

Exercise 04: PhoneBook

**This exercise will test Collections, Console IO, Control Statements, and Loops,**

Create an application to read data from a file. Each line in the file should have a First Name, Last Name, Street Address, Secondary Street Address (This may be blank), City, State, Country Code, Zip Code, Main Phone, Second Phone. Each data field should be comma delimited from the other fields.

Read each line into a Person object. The person object should have corresponding attributes to each field in the data record. You will need to create the person object outside of the PhoneBook class.

Create a main method in PhoneBook class which reads the data file and stores the data in a HashMap. Use the First Name as the key for the HashMap. Then create a menu to allow the user to exit the application, or search for a person.

If the user selects to search for a person, allow them to enter the first name and use it to search the HashMap. Note that HashMaps do not allow records with duplicate keys. If the record is found; return the information. If the record is not found, return a message letting the user know the message is not found.