PS 1 - CMPE 160.01: Introduction to Object Oriented Programming

Introduction to Eclipse and Java

04/03/2022

1 Introduction:

1.1 Java:

In this course, we will only program in Java. Some key points to consider are as follows:

- **What is the main difference between Java and Python?** Java is a compiled language whereas Python is an interpreted one.
- **What is a compiled language?** After the code is written, it is translated into machine code by a program called the compiler. Only then, the program can be run.

1.1.1 Java Naming Conventions:

Java naming conventions make Java programs more understandable and easier to read. Additionally, it may give information about the function of the identifier - whether it is a variable or class or function. It is not forced to follow since they are not rules but conventions.

- **Classes:** The class names should be nouns, with the first letter (as well as the first letter of each internal words) capitalized. Examples: class Student, class LinkedList
- **Methods:** The method names are generally verbs, with the first letter as lowercase and first letter of the internal words are capitalized. Examples: run(), getStudentNumber()
- **Variables:** The variable names start with a lowercase letter and should not start with underscore or signs even though they are allowed. Internal words may start with capital letters. One-character variable names should be avoided except the "temporary" variables. It is better to define mnemonic variable names. Examples: int i, float squareArea, String studentName
- **Packages:** The package names should be written in all lowercase letters.

1.2 Eclipse:

Eclipse is an integrated development environment (IDE) mainly for Java and other languages such as C++, Prolog, PHP, etc. In the course, you will be expected to get familiar with:

- Creating new projects
- Importing/exporting projects
- Running/debugging
- Writing/running JUnit test cases

1.2.1 Eclipse Views:

In Eclipse, you will see small windows called "views" Some of them are as follows:

- Package Explorer
- Console
- Teaching.Codes Main/Browser
- Problems

Do not get scared if you close windows. Just look at the menu bar: Window-¿Show View. Click "Other" for more views. There is also what is called perspective, which is a set of windows opened for your task. For different tasks such as coding, debugging, versioning, etc, there are different perspectives. **Hint:** In addition to Window-¿Show View for reopening the windows, you can use the reset functionality provided by the perspective. This can be also used in case of messing up with the locations of the windows.

1.2.2 Eclipse Details:

- **How to create a new project:** Click on the icon with the plus sign on a window on the icon bar $\longrightarrow$ Java Project $\longrightarrow$ Give it a name and proceed.

Alternatively, you could use menu bar or right click in package explorer.

- **How to create a new class:** Click on the name of your project in Package Explorer $\longrightarrow$ Click on the icon with the plus sign on letter C on the icon bar $\longrightarrow$ Make sure that project name is correct. $\longrightarrow$ Give it a name and proceed.

If you click the tiny arrow just next to the icon, you can create a new test case, interface, etc Alternatively, you could use menu bar or right click in package explorer.

- **How to run a project:** Open to the file that you want to execute. $\longrightarrow$ Click on the green button that looks like "play button"

Alternatively, you can right click the .java file, Run as Java Application.

1.2.3 Debugging:

- **What is debugging:** When projects get bigger and bigger, it becomes much easier to make mistakes. Debugging is the process of finding out bugs in the code.
- **Debugging in Eclipse:** Now, let's say your code is buggy and you certainly need to debug your code. The first thing to do is to set breakpoints. As its name implies, breakpoints are where execution is stopped and debugging begins

If you have an idea about the broken part in your code: Go to this line and double click to the line number at the beginning of the line. You will see a blue dot, this is a breakpoint.

If you do not have an idea about the broken part: Go the first line to be executed in the whole program and double click to the line number at the beginning of the line. You will see a blue dot, this is a breakpoint.

Then, find the bug in the icon bar (just next to run button) and click on. Accept if your are asked to switch perspective or not.

Most commonly used proceeding buttons:

- **Step Into(F5):** The current statement is executed. If it invokes some other methods, you would step into the execution of this method.
- **Step Over(F6):** The current statement is executed and switched to the next statement in your code.
- **Step Return(F7):** The current statement is executed until it returns something.
- **Resume(F8):** Run the program until the next breakpoint.

2 Exercises:

2.1 Checking Palindromes:

We wanted o create an example program that checks whether a given string is a palindrome or not.

A string is a palindrome if it reads the same forward and backward. The words "mom," "dad," and "noon," for instance, are all palindromes. The problem is to write a program that prompts the user to enter a string and reports whether the string is a palindrome. One solution is to check whether the first character in the string is the same as the last character. If so, check whether the second character is the same as the second-to-last character. This process continues until a mismatch is found or all the characters in the string are checked, except for the middle character if the string has an odd number of characters.

Example outputs are given below:

```
Enter a string: noon
noon is a palindrome
```

```
Enter a string: abcdefghikl
abcdefghikl is not a palindrome
```

Solution can be seen in 'Palindrome.java' file.

2.2 Displaying Prime Numbers:

Second example presents a program that displays the first 50 prime numbers in 5 lines, each line containing 10 numbers.

An integer greater than 1 is prime if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not. The problem is to display the first 50 prime numbers in 5 lines, each of which contains 10 numbers. The problem can be broken into the following tasks:

- Determine whether a given number is prime.
- For number = 2, 3, 4, 5, 6, ..., test whether it is prime.
- Count the prime numbers.
- Display each prime number and display 10 numbers per line.

Obviously, you need to write a loop and repeatedly test whether a new number is prime. If the number is prime, increase the count by 1. The count is 0 initially. When it reaches 50, the loop terminates.

Output of the program should be as follows:

```
he first 50 prime numbers are
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113
127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229
```

Here is the algorithm for the problem:

```
Set the number of prime numbers to be printed as
 a constant NUMBER_OF_PRIMES;
Use count to track the number of prime numbers and
 set an initial count to 0;
Set an initial number to 2;
while (count < NUMBER_OF_PRIMES) {
 Test whether number is prime;
 if number is prime {
 Display the prime number and increase the count;
 }
 Increment number by 1;
}
```

To test whether a number is prime, check whether it is divisible by 2, 3, 4, and so on up to number/2. If a divisor is found, the number is not a prime. The algorithm can be described as follows:

```
Use a boolean variable isPrime to denote whether
 the number is prime; Set isPrime to true initially;
for (int divisor = 2; divisor <= number / 2; divisor++) {
 if (number % divisor == 0) {
 Set isPrime to false
 Exit the loop;
 }
}
```