JAVA PROGRAMMING

Tutorial 03

Activity 1: Java Parser

Given below is a program named **JavaParserTest** that uses a java parser tool, named **`JavaParser`** to parse the program text of the class **`Hello`** and produce the standard Java source code text. The in-line comments give information about how to use **JavaParser**.

```
1  import com.github.javaparser.JavaParser;
2  import com.github.javaparser.ast.CompilationUnit;
3
4  public class JavaParserTest {
5    public static void main(String[] args) {
6        // program text
7        String progText = "class Hello { "
8        + "public static void main(String[] args) { "
9        + " System.out.println(\"Hello world!\"); "
10       + "}}";
11       // parse the program text
12       CompilationUnit codeUnit =
JavaParser.parse(progText);
13       // obtain the generated source code
14       System.out.println(codeUnit);
15   }
16 }
```

Do the following tasks:

- Run this program to produce the source code that is listed immediately below the program.

  Note: You need to:

  - Add to the class path the library file named javaparser-core-3.0.1.jar (provided in the resources/lib folder)

- Use import statements to import the necessary classes into your program.

- Change the program text (i.e. value of the variable **`progText`**) by removing the semi-colon (';') at the end of the print statement. Re-run the program and record what happened.
- Change the program text by removing the word "main". Re-run the program and record what happened.

Sample output:

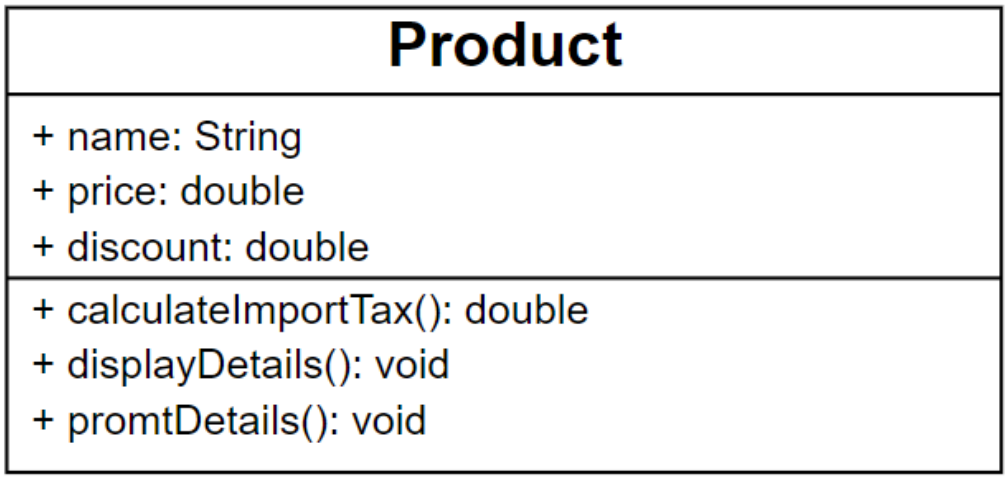
```
class Hello {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Activity 2: Product.java

Create a class named **`Product`** with three attributes: **name**, **price**, and **discount**. The class should also be able to: calculate import tax (10% of the product price) and display information on the screen.

Here is a sample class design of the Product:

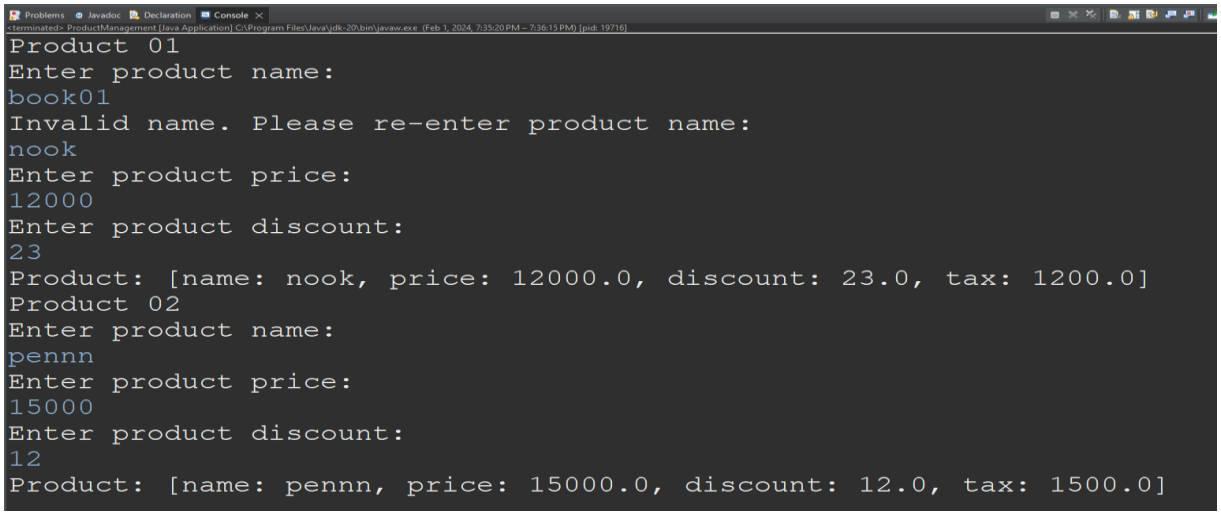


Activity 3: Product Management

In this activity, you will work with a program that uses a class to manage information about products. Complete the following steps to solve the task:

- **Create Product class:**
  - Start with the **`Product`** class (which has already been implemented). This class should have methods for entering and displaying product information.
- **Main method:**
  - Within the main method of your program, create two **`Product`** objects that you can name as you wish.
- **Enter information:**
  - Call the **`promtDetails()`** method to enter product data using the keyboard. **Note:** Please validate data before saving it to the object.
- **Display information:**
  - Call the **`displayDetails()`** method to display the product data on the screen.

Sample output:



Activity 4: Product Management (Upgraded)

In this activity, you'll enhance the **`Product`** class from the previous exercise. Follow these steps to make your improvements:

- **Modify Access Modifiers:**
  - Make the **`displayDetails()`** method `public`.
  - Make the **`calculateImportTax()`** method `private`.
- **Add Constructors:**
  - Introduce two constructors for the Product class.
    - ✓ Constructor 1: Accepts three parameters - **name**, **price**, and **discount**.
    - ✓ Constructor 2: Accepts two parameters - **name** and **price** (assuming no discount).
- **Implement Tests:**
  - Write a program to create two instances of the **`Product`** class.
  - One instance should have a discount, using the first constructor.
  - The other instance should have no discount, using the second constructor.
- **Display Information:**
  - Display the details of both products on the screen.

Activity 5: Employee Management (Optional)

Imagine a company with N employees, labeled Employee 1 through N. There are two tasks, Work A and Work B, that need to be finished. Each employee can complete Work A in a certain amount of time (Ai minutes) and Work B in another amount of time (Bi minutes).

You have the flexibility to assign both tasks to the same employee or to different employees. If both tasks are assigned to the same person, the time needed is simply the sum of the individual time for each task. However, if the tasks are given to different employees, it will take longer to complete both works.

The challenge is to figure out the shortest possible time required to complete both tasks efficiently. Can you find the optimal way to assign the tasks to minimize the overall time taken? Give it a try!

Constraints: [2 ≤ N ≤ 1000 1 ≤ Ai ≤ 100 1 ≤ Bi ≤ 100]

All values should be integers.

Input file format: data.txt

```
N
A1 B1
A2 B2
A3 B3
...
AN BN
```

Example 01:

Input: data.txt

```
3
6 5
4 2
6 6
```

Output: 5

Explanation: To efficiently complete tasks, let's consider assigning Work A to Employee 2 and Work B to Employee 1. Employee 2 can complete Work A in 4 minutes, while Employee 1 can finish Work B in 5 minutes. When assigning different tasks to different employees, the total time required is determined by the longer duration, which is 5 minutes in this case.

So, by assigning Work A and Work B to Employee 2 and Employee 1, respectively, the total time needed for both tasks to be finished is 5 minutes. It's important to note that when tasks are assigned to different individuals, we consider the maximum time required among them. In this scenario, it is not possible to complete the combined tasks in less than 5 minutes. Therefore, the minimum time required for completion is 5 minutes.

Example 02:

Input: data.txt

```
3
6 5
2 2
4 6
```

Output: 4

Explanation: For the most efficient results, consider assigning both tasks to Employee 2. Keep in mind that if you assign both tasks to the same person, the total time for completion is simply the sum of the times it takes for that person to complete each task individually.

This approach ensures optimal use of resources and helps streamline the workflow. Make sure to distribute tasks wisely for better efficiency!

Submission

Submit a zip file containing all Java programs to this tutorial's submission box in the course website on FIT Portal.