COMP 1030
Lab #6

Static Methods – Method Overloading

Introduction

During this lab you will build two java classes. The first class will contain the required state and behaviour for the object **but NO main method**. The second class will contain simply the main method to give the JRE an entry point into the program, a line to instantiate a new object based upon the first class and a few lines to exercise the functionality of the first class.

When writing your code, keep these guidelines in mind:

- Start each class with the proper javadoc comment header. The first line of that comment should be the purpose of the class not just its name.
- Provide a comment for **each** speciality method.
- Follow the layout for your class as illustrated below:

  Javadoc comment header
  Import statements (if required)
  Class declaration
    State (instance variables/data)
    Constructor(s) (if required)
    Behaviour(s) (method(s))
  Close class declaration

- Use whitespace and indentations to make your code more readable and easy to debug.
- Be sure to clearly understand your work – do not simply copy code from someone else.

Instructions: Use any IDE to complete this lab other than a simple notepad or BlueJ.

Step 1 Write a java class that adheres to the following criteria:

- Create a class called MovieTicket that will serve as a blueprint for a movie ticket. (**therefore contains no main method**).
  - This class should have:
    - The following State: movie name, ticket number, theatre number
    - Setters and getters for each instance variable
    - A static variable called ticketPrice **with a hardcoded initial value**.

  - A static method which calculates and returns the tax $ amount on the ticket price (assume 13% tax).
  - Two non-static overloaded methods which will return the movie run dates, one based upon the movie name and one based upon the ticket number. Use the switch statement to hardcode 4 different options for each method based upon data you make up.

- Create a second class called MovieTicketTestHarness which contains a **main method** to test the first class:
  - **Note: Any interaction with the user must be accompanied with an appropriate message.**
  - Instantiate a ticket.
  - Capture information from the user, and use the setters to populate all of the non-static fields of the ticket.
  - Print out all the fields of the ticket plus the price of the ticket and the tax portion of the ticket.
  - Ask the user for a movie name, use the appropriate method to determine and print the movie run dates.
  - Ask the user for a ticket number, use the appropriate method to determine and print the movie run dates.
  - **Note: Any interaction with the user must be accompanied with an appropriate message.**

Things to consider for success:

- Follow the layout for your class as illustrated above.
- Write a section at a time and compile after each section so you do not have a volume of compiler errors to deal with.
- Comment as you go.
- Use indents and whitespace appropriately to make your code readable.
- Be sure that you actually understand the code you have written, simply copying someone else's code will not aid in your understanding of the java language.
- Stay focused and work diligently, collaborate with others if you are stuck.