RMIT University - School of CSIT
CPT121 - Programming 1 (COSC2135)
Assignment 1 - Study Period 1, 2014

Introduction

You are required to implement a basic Java program using Java SE 1.6 / 1.7

This assignment is designed to:

- Test your knowledge of basic Java concepts
- Evaluate your ability to design programming logic
- Practise simple object design in Java.

This is an **individual assignment** and contributes **15%** towards your final grade.

Academic Integrity (more)

The submitted assignment must be your own work. No marks will be awarded for any parts which are not created by you.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools may be in cases where plagiarism is suspected. Any cases of plagiarism will be automatically referred to the school and the students involved will dealt with according to RMIT University policies regarding academic integrity. You can find more information on RMIT University policies and procedures relating to academic integrity at http://www.rmit.edu.au/browse;ID=sg4yfqzod48g1.

Preamble - Simple Toy Store Inventory System

For this task you will be implementing a console based program which focuses on object-oriented design in Java. The console based program will implement the basic Toy Store scenario as discussed below.

The management of inventory in the Toy Store needs to be able to cater for receiving deliveries of stock from suppliers, selling stock to customers, and displaying the current status of all toys (toys) that the Toy Store stocks.

There are also features which staff can use to display a list of items that need to be restocked and list all products in a specified range via keyword search.

Toy class (35 marks)

For each Toy (item) that the Toy Store stocks the system maintains details for the product code, toy description, unit price and current stock level.

You will be required to implement a class **Toy** which encapsulates the details for a Toy, as well as implementing the required functionality that is described below:

- Define private instance variables representing the Toy details mentioned above. **(5 Marks)**
- Implement a constructor which initialises the instance variables for a new Toy object, given the product code, description, unit price and initial stock level as parameters. **(5 Marks)**
- Implement accessors for each of the instance variables described above. **(5 Marks)**
- Implement a method `public double sell (int units)`, which updates the stock level and returns the total sale cost when a customer purchases one or more units of a Toy.

  If the number of units requested exceeds the current stock level then the operation should fail and a result of **Double.NaN** should be returned, otherwise the current stock level for the Toy should be updated to reflect the sale and the total sale cost (ie. **unit cost * quantity**) should be returned. **(10 Marks)**
- Implement a method `public void restock(int units)`, which adds the specified number of units to the current stock level for a Toy when new stock of the Toy in question is received from a supplier.

  No validation of the amount of stock being "delivered" is required and nothing needs to be returned for this method. **(5 marks)**
- You should also include one of the following helper methods to assist with the displaying of the current state of a Toy object to the screen (you only need to do one of these):
  - A method `public void printDetails()`, which displays the details for a Toy object to the screen in a neat, tabulated manner (using System.out.printf()) **(5 Marks)**
  - An implementation of the method `public String toString()` which returns the details for the Toy as a single String which is formatted in a neat, tabulated manner. You do not need to display anything to the screen at this point.

    If you choose this option then the method Java API method `String.format()` will allow you to construct a String which is formatted in a manner similar to how `System.out.printf()` allows you to format output that is being displayed to the screen. **(5 Marks)**

You are required to adhere to all recommended object-oriented principles (ie. encapsulation, information hiding, etc) when implementing this class.

The ToyStore class (55 marks)

Once you have completed the implementation for the Toy class shown above you will be required to demonstrate how to create and use a collection of objects created from the class in a Java program (application).

This ToyStore application class should implement a menu-driven console program which allows the user to repeatedly select their desired option until they choose to exit the program.

Your program will be required to implement the following functionality:

a) Creating and storing Toy objects (8 marks)

An array has already been defined in the ToyStore class which can store up to eight (8) Toy objects - populate this array with a set of eight (8) Toy objects containing the (hard-coded) details shown in the table below:

Product code	Description	Unit Price	Stock level
LEG-301	Lego Star Wars Super Pack	$119.99	2
LEG-210	Lego Friends Sunshine Ranch	$79.99	1
TOY-005	Toy Story - Talking Sheriff Woody	$54.99	12
NER-020	Nerf Rapid N-Strike Kit	$38.00	15
HOT-101	Hot Wheels Cascade Blitz	$99.99	10
LEG-401	Lego Movie Evil Business Lord's Lair	$69.99	10
LEG-420	Lego Movie Clock - Bad Cop	$24.99	0
TOY-008	Toy Story - Jessie the Talking Cowgirl	$39.99	15

b) Program menu (already implemented)

A program menu has been implemented which displays the following options to the screen.

```
Toy Store Inventory Program
---------------------------

Display Inventory       A
Print Reorder List      B
Product Range Search    C
Accept Delivery         D
Make Sale               E
Exit Program            X

Enter selection:
```

You will need to complete the implementation of each of the features in this menu driven program as described below.

c) Display Inventory Feature (5 marks)

This feature should display the details for each Toy in the Toy Store's inventory in a tabular form, by either stepping through the array of Toy objects (using a suitable loop) and invoking either the `printDetails()` method -OR- invoking the `toString()` method and printing the result it returns for each object in the array (depending on which one you implemented in your Toy class).

d) Print Reorder List (6 marks)

This feature should locate all toys for which the current stock level is less than five (5) units and display the product code, description and stock level for each toy in question.

You must use a valid search technique to locate all Toy objects which meet this criteria (ie. use a loop to iterate through the array and then check each Toy object to see if it meets the criteria discussed above).

e) Product Range Search feature (6 marks)

This feature should allow the user to perform a "keyword" search for items in a particular product range - eg. if the user enters "Lego Movie" then all Toys which have the keyword "Lego Movie" as part of their description should be returned.

If one or more matches are found then the feature should display the description, unit price and availability (ie. "in stock" or "out of stock") for each matching Toy that is found. If no matching Toy objects were found then a suitable error message should be displayed to the screen indicating that they keyword search didn't return any matching results.

You must use a valid search technique to locate all Toy objects which meet this criteria (ie. use a loop to iterate through the array and then check each Toy object to see if it meets the criteria discussed above) - you cannot rely on Toy objects with specific descriptions being stored at specific positions in the array for this feature.

f) Make Sale feature (18 marks)

This feature should first prompt the user to enter a product code to search for, after which it should attempt to locate a Toy with a matching product code from among the collection of Toy objects in the array.

You **must** demonstrate an appropriate search technique using a loop to step through each Toy object in the array and extract its product code for comparison with the product code entered by the user here, rather than relying on/assuming that the Toy objects are at specific positions.

If no Toy with the specified product code was found then sale should be rejected by displaying a suitable error message indicating that the specified product code was not found to the screen.

If a Toy with a matching product code is found then the user should be prompted to enter the number of units being purchased, after which the program should attempt to update the stock level for the Toy in question by invoking the **sell()** method in an appropriate manner for that Toy.

If this call to the **sell()** method for the Toy in question fails (ie. it returns a result of **Double.NaN**) then a suitable error message indicating that there is insufficient stock should be displayed to the screen. If the call to the **sell()** method for the Toy in question succeeds (ie. it returns a valid double result) then the program should trap the result that was returned and display it to the screen as the total sale cost.

Note that you will not be able to check for the value **Double.NaN** using the simple equality ('==') operator, so you will need to do a little research to work out how to check for this specific value in the Java API (or online).

g) Accept Delivery feature (12 marks)

This feature should first prompt the user to enter a product code to search for, after which it should attempt to locate a Toy with a matching product code from among the collection of Toy objects in the array.

You **must** demonstrate an appropriate search technique using a loop to step through each Toy object in the array and extract its product code for comparison with the product code entered by the user here, rather than relying on/assuming that the Toy objects are at specific positions.

If no Toy with the specified product code was found then the delivery should be rejected by displaying a suitable error message indicating that the specified product code was not found to the screen.

If a Toy with a matching product code found then the user should be prompted to enter the number of units being delivered, after which the program should update the stock level for the Toy in question by invoking the **restock()** method in an appropriate manner for that Toy.

The feature should then display message to the screen confirming the delivery has been accepted as well as the new stock level for the Toy to the screen.

Coding Style (5 Marks)

Your program should demonstrate appropriate coding style, which includes:

- Levels of 3 or 4 spaces used to indent rather than tabs - you can set up your IDE/editor to automatically replace tabs with levels of 3 or 4 spaces.
- Indentation levels used are consistent throughout program
- A new level of indentation added for each new class/method/ control structure used
- Going back to the previous level of indentation at the end of a class/method/control structure (before the closing brace if one is being used)
- Lines of code not exceeding 80 characters in length where possible - it is ok to go beyond this limit by a small amount occasionally when required, but lines which will exceed this limit by a significant amount or do so on a regular basis should be split into two or more segments where needed
- Expressions are well spaced out and source is spaced out into logically related segments
- Use of appropriate identifiers wherever possible to improve code readability
- Use of comments to describe the purpose of each class, each method within your classes and any non-trivial segments of code within those methods.

General Implementation Notes/Guidelines

You are being assessed on your ability to write a program in an object-oriented manner in this assignment - as such you should treat the Toy class implementation as if it is a blueprint for a single Toy in the ToyStore's inventory and demonstrate how you can create the required set of Toy objects in the ToyStore class and use/manipulate this set of Toy objects in different ways.

Your submission **will be penalised heavily** if you go against these guidelines by attempting to implementing the program in a procedural manner or treating the Toy class as a repository for all Toy information, instead of just a blueprint for a single Toy - **refer to the material covered in weeks 5-6 of the course to get a better understanding of the philosophy behind developing your program for this assignment**.

You should stick to using the **standard Java API** (**version 1.6 and 1.7 are both ok**) when implementing your program – use of third party API's will mean that the markers will not be able to compile and run your program, **which will result in significant penalties for a "non-functional" program being applied**.

If you are using the Scanner class for reading input then it is likely that you will run into the "Scanner use bug" at some points in the user-input sequence for this program – the input sequences were deliberately structured to create a potential "Scanner use bug" issue at different points and you should deal with this problem in an appropriate manner where required.

What to submit

If you have developed the program using a text editor and compiling/running manually at a command prompt then you only need to submit the **source code** (.java) files for the Toy class and your separate ToyStore application class (which should contain the main() method).

If you are using eclipse then you should export your entire eclipse project to a zip archive and submit the resulting zip file - **do this from within eclipse while it is running**, not by trying to copy or move files around in the eclipse workspace directly, as you may corrupt your entire workspace if you do something wrong.

All students are also advised to check the contents of their zip files by opening them and viewing the files contained within before submitting to make sure they have done it correctly and that the correct (latest) version of the source code file is present to avoid any unpleasant surprises later on – we are obliged to accept each submission in the form it is sent to us in, so make sure you submit the final version of your program!

Submission information

This assignment will be marked out of a total of **100 marks** and contributes **10%** towards your final result for this course. This assignment is not a hurdle in/of itself - rather it contributes towards the practical work component for this course (**of which you must obtain 50% or better overall in order to pass the course**).

This assignment is due to be submitted to **weblearn** by **11:59pm on Wednesday April 30 AEST - this is the middle of week 9.**

There will be **a late submission period of 5 days** for this assignment, which will expire at **11:59pm on Monday May 5 AEST**. Late submissions that are received before the end of this late submission period will **attract a late penalty of 10% per day (or part thereof) of the marks awarded on the assignment**, unless an extension has been granted by the school or as part of an existing EAA provision (see below for details).

Submissions that are received after the late submission period expires at 11:59pm on Monday May 5 AEST will not be assessed, unless some prior arrangement has been organised by the CSIT OUA administrator (ouacsit@rmit.edu.au) in response to a request for an extension.

Extensions

Your instructor does not have the authority to negotiate or approve general extensions, so all such requests should be sent to the email address listed above.

Note that you will normally be required to provide supporting documentation with your extension so having such documentation ready when you apply for an extension will streamline the extension request process.

Special Note for students with EAA provisions organised by the DLU

If you are a student who has EAA provisions organised by the DLU then you must negotiate any extension that you may need with the instructor well in advance of the deadline (at least 72 hours in advance of the on-time submission deadline would be preferred).

Further Information

All questions regarding this assignment should be directed to the appropriate Assignment 2 Discussion forum on blackboard.