EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Contents

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Instructions:

Important Reminders

- You should attend your lab session (the one you are enrolled in). If you need to change your lab enrollment, you should contact the Undergraduate Office in the department. ***Instructors or TAs cannot change your enrollment***.
- You can submit your lab work in eClass any time before 22:00 on Friday (**July 21, 2023**) of the week the lab is due. Your last submission will overwrite the previous ones, **and only the last submission will be graded**.
- The deadline is strict, with no excuses: **you receive 0 for not making your electronic submission in time**. ***Emailing your solutions to the instructors or TAs will not be acceptable.***
- To submit your work, you need to use the York eClass.
- **Your submission will be graded by JUnit tests given to you and additional JUnit tests covering some other input values.** ***This is to encourage you to take more responsibility for the correctness of your code by writing more JUnit tests.***
- Developing and submitting a correct solution for this lab without compilation errors is essential. Hence, you must take a reasonable amount of time to test your code in different ways. If you submitted a solution with a small mistake in terms of syntax or do not comply with lab instructions, then you may receive 0 as a grade for the implementation of this lab
- There will be **a 25% penalty** on your lab final grade if your submitted code does not compile due to ***minor compilation errors***, given that TAs can fix these minor compilation errors. **You will receive a zero if your code contains major compilation errors that TAs can not fix**.

Academic Honesty

- Students are expected to read the Senate Policy on Academic Honesty. See also the EECS Department Academic Honesty Guidelines.
- **All labs are to be completed individually: no group work is allowed. Do not discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.** ***If you are repeating the course, you are not allowed to submit your own solution developed in previous terms or for other purposes. You should start from scratch and follow the instructions.***

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Learning Outcomes and Objectives

The lab aims to familiarize students with creating new data types using a class. Your task is to write the Java code that meets these classes' specifications. Also, test your Java code using JUnit testing.

Learning Outcomes

- **CLO1**: *Create new data types using the notion of a class, declared with primitive and/or reference attributes, constructors, and methods (accessors and/or mutators).*
- **CLO2**: *Illustrate the difference between a class and its instances (objects) by writing a program which instantiates objects from classes and calls methods on those objects.*
- **CLO3**: *Document classes (with e.g., UML class diagrams, JavaDoc) and methods (with preconditions and assertions).*
- **CLO4**: *Write unit tests to check correctness of classes and use an IDE debugger to correct errors.*
- **CLO5**: *Use primitive arrays*, linked lists, and library collections (e.g., lists, tables) to implement iterative and recursive algorithms including searching (e.g., linear vs. binary) and sorting (e.g., selection sort, insertion sort, merge sort).

Lab Learning Objective

- To create a class with some attributes.
- To create different types of constructors; default, copy, ...
- To use a chain of constructors
- To create an object and call setter or getter methods
- Declaring and manipulating (single-valued vs. multi-valued) reference-typed attributes
- To use Use Java controls structure (selection structures, repetition structures, and nested Loops)
- To be familiar with using Arrays: 1D and 2D
- To use String data type and its methods
- To use JUnit Tests to verify your work

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Getting Started

1. Start eclipse.
2. **Download the starter code "Lab7.zip" from the eClass course site**
3. Import the test project by doing the following:
    1. Under the **File** menu, choose **Import...**
    2. Under **General,** choose **Existing Projects into Workspace** and press **Next**
    3. Click the **Select archive file** radio button, and click the **Browse...** button. You may have to wait about 10 seconds before the file browser appears.
    4. In the file browser that appears, navigate to your home directory.
    5. Select the file **Lab7.zip** and click **OK**
    6. Click **Finish**.
4. All files you need for this lab should now appear in eclipse.

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Lab Structure

After successfully importing the starter code/project "Lab7.zip"

The lab folder/directory structure is as follows:

- **src/lab7**/: directory contains Java files: Your task is to determine the number of classes needed for this lab and to specify their names. Check the **JunitTest_ChildOrderToyTest.java.**
- **src/lab7**/: directory contains Java files (JUnit test cases): **JunitTest_ChildOrderToyTest.java.** These files contain several JUnit test cases that can help to test your code.
  It should be noted that you need to run the JUnit tester **JunitTest_ChildOrderToyTest.java** *after you complete the lab to check your work. Nonetheless, passing all given tests does not guarantee full marks for this lab. Therefore, you are required to write additional tests to ensure the correctness of your implementations.*

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Lab Restrictions:

- Any use of Java library classes or methods (e.g., **ArrayList, System.arraycopy**) is forbidden. That is, there must **not** be any import statement at the beginning of this class. Violation of this requirement will result in a **70% penalty** on your marks.
- For the JUnit test cases class **JunitTest_ChildOrderToyTest.java** given to you
  - Do not modify the test methods given to you.
  - You are allowed to add new test cases by creating new test methods.
- For each method which you are required to implement, **derived from the JUnit test methods**:
  - No **`System.out.println`** statements should appear in it.
  - No Scanner operations (e.g., **`input.nextInt()`**) should appear in it. Instead, declare the method's input parameters as indicated by the JUnit tests.
- **Hint: You may use `java.util.Arrays` class, class `Character` or Class `String`.**

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

Lab Exercise

Computational thinking for a software developer/computer programmer is a critical skill that is consistently applied. This lab requires you to develop a solution using Java object-oriented programming that simulates an order system for children's toys. The child will manage a collection/list of toys (***each child owns his toys***), and an order system will create an order containing up to **5 children** (i.e., the maximum number of children in any given order is 5).

The system manages information about a single toy. The system stores the following information (for each attribute, choose any type that you think is appropriate--you must be able to justify the decisions you make):

- Toy ID: the toy id of the toy (a positive number)
- Toy name: the name of the toy
- Toy quantity: the quantity of this toy that the child owns. The quantity of any given toy is a positive integer value.
- Toy price: the price of this toy

The system manages information about a single child. The system stores the following information (for each attribute, choose any type that you think is appropriate--you must be able to justify the decisions you make):

- Child name: the name of a child
- Child age: a positive integer number represents the child's age in years.
- Child list of toys: **the system allows a child to have as many toys as the child want**. There is **no maximum** number of toys that a child can own.
- Child number of toys: a positive number represents a child's number of toys at any given moment.

EECS1022 -Winter 2023- Lab07

Due Date: Friday, July 21, 2023, before 22:00

The system manages information about a single order. The system stores the following information (for each attribute, choose any type that you think is appropriate--you must be able to justify the decisions you make):

- List of children in this order: **The maximum number of children in any given order is 5**. Thus, an order can contain any number of children between **zero and 5**.
- The number of children: the order object stores the number of children included in this order at any given moment.

Your task is to drive and create class(es) and method(s) from the given JUnit class and then add them to the package using the following facts:

- You can get and set any toy attributes.
- The child object can be constructed by receiving the **child's name**, **age** and **list of toys**.
- **Each child owns their toys and does not share them with other children.**
- A child can dispose of his toys.
- A child can denote his toys only to another child. In this case, the toys will be removed from the donor child, and the toys will be added to the list of toys of another child. There is no need to check for **duplicated toys** when denoting child toys (e.g., two toys with the same information/names).
- The system allows you to add toys to the list of toys for any child, ensuring that the toys **are not duplicated based on their names** and using the existing ID if the toys have different IDs (i.e., ignore the new toy ID). **Ensure you update the toy quantity and price by considering the maximum price for the first occurrence of duplicate toys.**
- You can get and set the child's name.
- You can get and set the child's age.
- You can get and set the child's list of toys.
- You can add a child to the order as long as you do not reach the maximum number of children per order, which is 5.
- You can remove a child from the order. If the child exists in a given order, then you need o update the number of children in this order. Otherwise, do nothing.
- You can retrieve the list of children in any given order.

Note: Any additional system functionalities can be inferred from the JUnit test.

Note: It is expected that the **JunitTest_ChildOrderToyTest.java** JUnit class contains compilation errors. This is because the declarations and definitions of the required class(es) and method(s) it references are missing. It is essential to drive and create class(es) and method(s) from the given JUnit class and then add them to the package.

Infer Class and Method Specifications from JUnit Tests

You may have noticed that the above "overview" descriptions are not as precise as the class specifications and method implementations. In fact, unlike the previous labs, we will not provide detailed specifications in this handout. To obtain the precise specification, you need to carefully ***analyze the test cases*** in the provided JUnit tests to understand the expected behaviours of each method.

In professional software development, test cases often play a vital role in specifying software requirements. This is called **Test-Driven Development** (TDD). Read more about it here: https://en.wikipedia.org/wiki/Test-driven_development.

Submit your work by using the course eClass

Check List:

Before submitting your files for this lab, you need to make sure you completed the following

	There is No compilation error generated from your implementation.
	The **Three** files contain the implementation for this lab.

Submit The Following File:

1) You need to submit **Three** files.