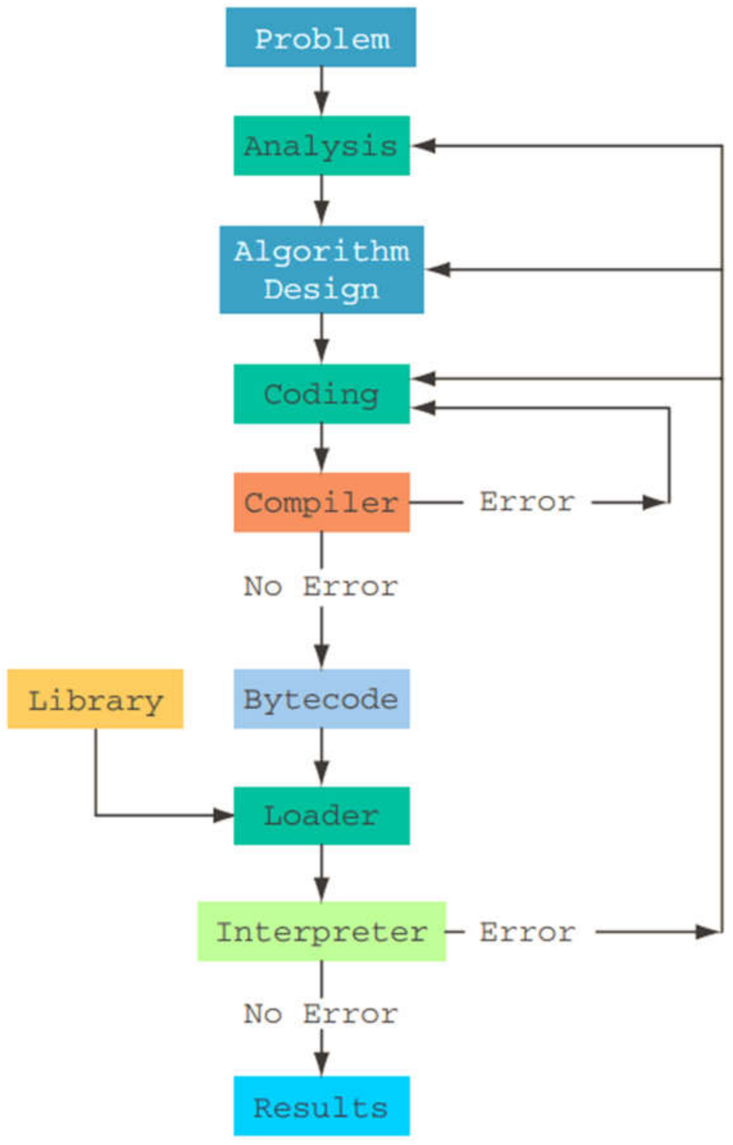
Deadline: 26-Sep-2023

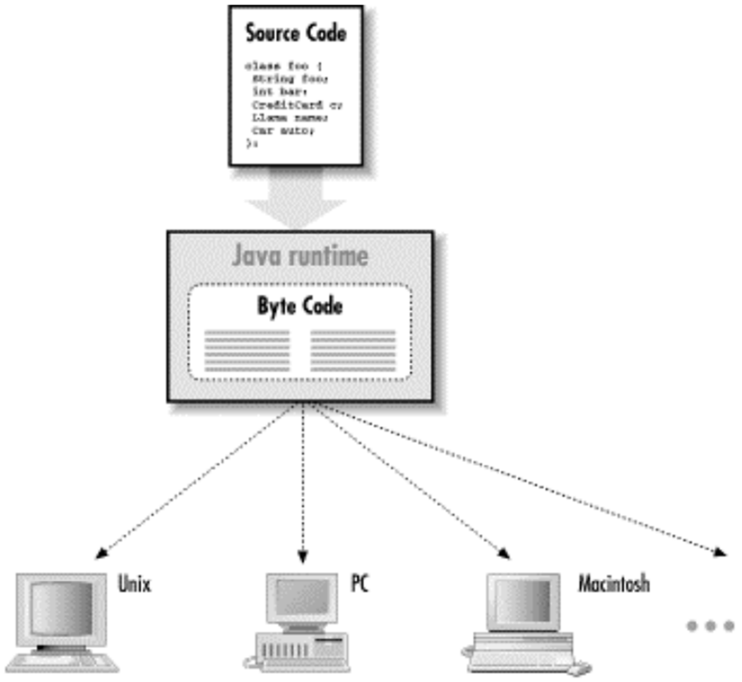
Lab 2: JAVA Conventions & Project Guidelines

CLO: 1, 2

Using the proper structure makes a Java program easier to understand and modify. It is frustrating trying to follow, and perhaps modify, a program that is syntactically correct but has no structure. Every Java application program must satisfy certain language rules. Syntax; the use of blanks; the use of semicolons, brackets, and commas; semantics; prompt lines; documentation, including comments and naming identifiers; and form and style.

The diagram summarizes process for Java Programming Language.





```
class foo {
  String foo;
  int bar;
  CreditCard cc;
  Llama name;
  Car auto;
};
```

E

Examples: Object Oriented Design for Problem Statements

"Write a **program** to *input* the **length** and **width** of a **rectangle** and *calculate* and *print* the **perimeter** and **area** of the **rectangle**."

Step 1: Identify all the (relevant) nouns.

- Length
- Width
- Perimeter
- Area
- Rectangle

Step 2: Identify the class(es).

Considering all five nouns, it is clear that:

- Length is the length of a rectangle.
- Width is the width of a rectangle.
- Perimeter is the perimeter of a rectangle.
- Area is the area of a rectangle.

Notice that four of the five nouns are related to the fifth one, namely, `rectangle`. Therefore, choose `Rectangle` as a class. From the `class Rectangle`, you can instantiate rectangles of various dimensions. The `class Rectangle` can be graphically represented

Step 3: Identify the data members for each of the classes.

In this step, you evaluate the remaining nouns and determine the information that is essential to fully describing each class. Therefore, consider each noun—length, width, perimeter, and area—and ask: "Is each of these nouns essential for describing the rectangle?"

- Perimeter is not needed, because it can be computed from length and width. Perimeter is not a data member.
- Area is not needed, because it can be computed from length and width. Area is not a data member.
- Length is required. Length is a data member.
- Width is required. Width is a data member.

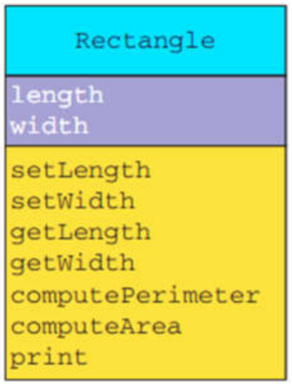
Step 4: Identify the operations for each of the classes.

Many operations for a class or an object can be determined by looking at the list of verbs. Let us consider the verbs *input*, *calculate*, and *print*. The possible operations on a rectangle object are *input* the length and width, *calculate* the perimeter and area, and *print* the perimeter and area. In this step, we focus on the functionalities of the class(es) involved. By carefully reading the problem statement, you may conclude that you need at least the following operations:

- **`setLength`**: Set the length of the rectangle.
- **`setWidth`**: Set the width of the rectangle.
- **`computePerimeter`**: Calculate the perimeter of the rectangle.
- **`computeArea`**: Calculate the area of the rectangle.
- **`print`**: Print the perimeter and area of the rectangle.

It is customary to include operations to retrieve the values of the data members of an object. Therefore, you also need the following operations:

- **`getLength`**: Retrieve the length of the rectangle.
- **`getWidth`**: Retrieve the width of the rectangle.



With these steps completed, you can design an algorithm for each operation of an object (class) and implement each algorithm in Java.

Consider the following problem:

A **place** to buy **candy** is from a **candy machine**. A new candy machine is purchased for the **cafeteria**, but it is not working properly. The candy machine has four **dispensers** to hold and release **items** sold by the candy machine as well as a **cash register**. The machine sells four products—**candies**, **chips**, **gum**, and **cookies**—each stored in a separate dispenser. You have been asked to write a program for this candy machine so that it can be put into operation.

The program should do the following:

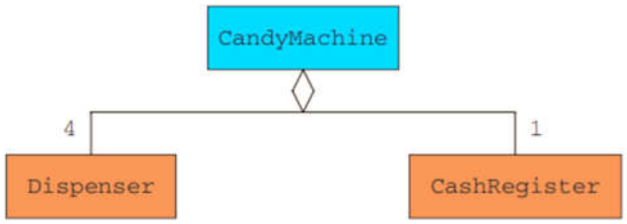
- *Show* the **customer** the different **products** sold by the **candy machine**.
- Let the **customer** *make* the selection.
- *Show* the **customer** the **cost of the item** selected.
- *Accept* the **money** from the **customer**.
- *Return* the **change**.
- *Release* the **item**, that is, *make* the sale.

The OOD solution to this problem proceeds as follows:

Step 1: Identify all the nouns.

Place, **candy**, **candy machine**, **cafeteria**, **dispenser**, **items**, **cash register**, **chips**, **gum**, **cookies**, **customer**, **products**, **cost** (of the item), **money**, and **change**.

In this description of the problem, products stand for items such as candy, chips, gum, and cookies. In fact, the actual product in the machine is not that important. What is important is to note that there are four dispensers, each capable of dispensing one product. Further, there is one cash register. Thus, the candy machine consists of four dispensers and one cash register.



Step 2: Identify the class(es).

You can see that the program you are about to write is supposed to deal with dispensers and cash registers. That is, the main objects are four dispensers and a cash register. Because all the dispensers are of the same type, you need to create a class, say, `Dispenser`, to create the dispensers. Similarly, you need to create a class, say, `CashRegister`, to create a cash register. We will create the `class CandyMachine` containing the four dispensers, a cash register, and the application program.

Step 3: Identify the data members for each of the class(es).

Dispenser To make the sale, at least one item must be in the dispenser and the customer must know the cost of the product. Therefore, the data members of a dispenser are:

- Product cost
- Number of items in the dispenser

Cash Register The cash register accepts money and returns change. Therefore, the cash register has only one data member, which we call `cashOnHand`.

Candy Machine The `class CandyMachine` has four dispensers and a cash register. You can name the four dispensers by the items they store. Therefore, the candy machine has five data members—four dispensers and a cash register.

Step 4: Identify the operations for each of the objects (classes).

The relevant verbs are *show* (selection), *make* (selection), *show* (cost), *accept* (money), *return* (change), and *make* (sale).

The verbs *show* (selection) and *make* (selection) relate to the candy machine. The verbs *show* (cost) and *make* (sale) relate to the dispenser. Similarly, the verbs *accept* (money) and *return* (change) relate to the cash register.

Dispenser The verb *show* (cost) applies to either printing or retrieving the value of the data member `cost`. The verb *make* (sale) applies to reducing the number of items in the dispenser by 1. Of course, the dispenser has to be nonempty. You must also provide an operation to set the cost and the number of items in the dispenser. Thus, the operations for a dispenser object are:

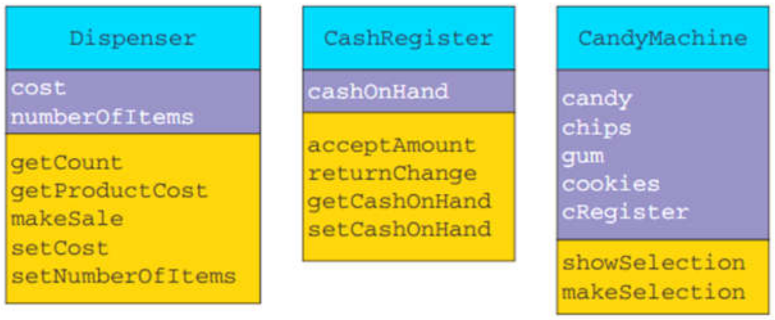
- `getCount`: Retrieve the number of items in the dispenser.
- `getProductCost`: Retrieve the cost of the item.
- `makeSale`: Reduce the number of items in the dispenser by 1.
- `setCost`: Set the cost of the product.
- `setNumberOfItems`: Set the number of items in the dispenser.

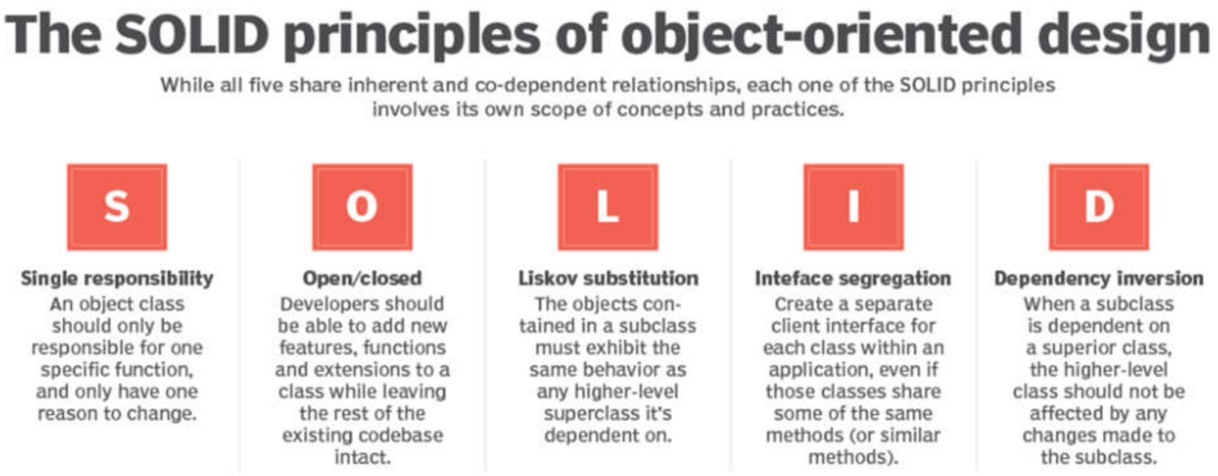
Cash Register The verb *accept* (money) applies to updating the money in the cash register by adding the money deposited by the customer. Similarly, the verb *return* (change) applies to reducing the money in the cash register by returning the overpaid amount (by the customer) to the customer. You also need to (initially) set the money in the cash register and retrieve the money from the cash register. Thus, the possible operations on a cash register are:

- **acceptAmount**: Update the amount in the cash register.
- **returnChange**: Return the change.
- **getCashOnHand**: Retrieve the amount in the cash register.
- **setCashOnHand**: Set the amount in the cash register.

Candy Machine The verbs *show* (selection) and *make* (selection) apply to the candy machine. Thus, the two possible operations are:

- **showSelection**: Show the number of products sold by the candy machine.
- **makeSelection**: Allow the customer to select the product.




The SOLID principles of object-oriented design
While all five share inherent and co-dependent relationships, each one of the SOLID principles involves its own scope of concepts and practices.
S
Single responsibility
An object class should only be responsible for one specific function, and only have one reason to change.
O
Open/closed
Developers should be able to add new features, functions and extensions to a class while leaving the rest of the existing codebase intact.
L
Liskov substitution
The objects contained in a subclass must exhibit the same behavior as any higher-level superclass it's dependent on.
I
Inteface segregation
Create a separate client interface for each class within an application, even if those classes share some of the same methods (or similar methods).
D
Dependency inversion
When a subclass is dependent on a superior class, the higher-level class should not be affected by any changes made to the subclass.

Deadline: 26-Sep-2023

Task

CLO: 1

- Describe the problem domain, significance and business rules.
- Describe Functional Requirements (User Requirements and System Specification)
- Describe Non-Functional Requirements or Design goals
- Identify classes
- Identify relationships (association, aggregation, composition) among entities
- Identify cardinalities/multiplicities

Update you project proposal with detailed problem statement, analysis and design. Make the UML diagrams as depicted in examples.