Creating Objects

Part a - Coin Toss

Write a class called Coin. The Coin class should have the following field:

A String named sideUp. The sideUp field will hold either "heads" or "tails" indicating the side of the coin that is facing up.

The Coin class should have the following methods:

A no-arg constructor that randomly determines the side of the coin that is facing up ("heads" or "tails") and initializes the sideUp field accordingly.
A void method named toss that simulates the tossing of a coin. When the toss method is called, it randomly determines the side of the coin that is facing up ("heads" or "tails") and set the sideUp field accordingly.
A method named getSideUp that return the value of the sideUp field.

Write a program that demonstrates the Coin class. The program should create an instance of the class and display the side that is initially facing up. Then, use the a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up. The program should keep count of the number of times heads is facing up and the number of times the tails is facing up, and display those values after the loop finishes.

Part b - Parking Ticket Simulator

Create a set of classes simulating a police officer issuing a parking ticket.

Class - ParkedCar has the following responsibilities:
To know the car's make, model, color, license number and number of minutes the car has been parked.

Class - ParkingMeter
This class should simulate a parking meter. This class's only responsibility is as follows:
To know the number of minutes of parking time that has been purchased.

Class - ParkingTicket has the following responsibilites:
To report the make, model, color and license number of an illegally parked car.
Report the fine - which is $25.00 for first hour plus $10.00 for each additional hour
Report the name and badge number of the police officer issuing the ticket.

Class - PoliceOfficer has the following responsibilities:

To know the name and badge number

To examine a parked car and parking meter object and determine whether the car's time has expired.

To issue a parking ticket if the car time has expired.

Plan of attack

Coin Toss Simulator

Learning objectives

After completing this assignment you will be able to:
1. Create Classes using Java coding standards and Object Oriented Programming basics.
2. Analyze requirements, design, code and test small Java programs.
3. Apply Association relationship in design.

Understanding requirements

Now lets start with understanding requirements. You are being asked to build a coin toss simulation program. The simulation program should toss coin randomly and track the count of heads or tail.

Let's think about what needs to be built.
We need to write a program that can perform following operations:
a. Toss a coin randomly
b. Track the count of heads or tails
c. Display the results.

Designing

Let's decide what classes, methods and variables will be required in this task and their significance.
1. Write a class called Coin.
2. The Coin class should have an Instance variable or an Enumerator named sideUp. The sideUp field will hold either "heads" or "tails" indicating the side of the coin that is facing up.
3. The Coin class should have following methods:
    a. A void method named toss that simulates the tossing of a coin. When the toss method is called, it randomly determines the side of the coin that is facing up ("heads" or "tails") and set the sideUp field accordingly.
    b. A no-arg constructor that randomly determines the side of the coin that is facing up ("heads" or "tails") and initializes the sideUp field accordingly.
    c. A method named getSideUp that return the value of the sideUp field.
4. This Coin class should be operated by an external wrapper class Simulation.
    - The program should create an instance of the class and display the side that is initially facing up.
    - Then, use the a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up.
    - The program should keep count of the number of times heads is facing up and the number of times the tails is facing up, and display those values after the loop finishes.

5. Write the test program that demonstrates the Simulation class. Handle different cases possible.

Topics to Learn
---

1. Object Oriented Programming
2. Association in design
3. Math.random() function
4. Testing including boundary conditions
5. UML Class diagram

****

Parking Ticket Simulator

Learning objectives
---
1. Analyze requirements, design, code and test Java programs

Understanding requirements

Design to simulate a system where a police officer issues a parking ticket. You should design a set of classes that work together as a Parking Simulator.

Designing

Similar to the approach followed in the previous task, let's decide what classes, methods and fields should be required.

Since the PoliceOfficer will make the decision on whether to issue ticket, issueTicket method belongs to PoliceOfficer. ParkedCar, ParkedTicket and ParkedMeter are independent entities described below.

ParkedCar Class has the following responsibilities:

- To know the car's make, model, color, license number and number of minutes the car has been parked. Create getter and setter methods for all the fields.

ParkingMeter Class only responsibility is as follows:

- To know the number of minutes of parking time that has been purchased.

ParkingTicket Class has the following responsibilities:

- Report the make, model, color and license number of an illegally parked car.
- Report the fine - which is $25.00 for first hour plus $10.00 for each additional hour
- Report the name and badge number of the police officer issuing the ticket.

PoliceOfficer Class has the following responsibilities:

- To know the name and badge number
- To examine a parked car and parking meter object and determine whether the car's time has expired.
- To issue a parking ticket if the car time has expired.

Write the test classes including testing situations:
1. ParkedVehicle is with in, out of and just in the parking time purchased.
2. Ticketing under 1 hour and more than 1 hour test cases.

Topics to Learn
---
1. Object Oriented Programming
2. Association in design
3. Class diagram

Submitting your work

1. Check each part of your submission against the following grading criteria.
    a. Your design satisfies the criteria of using pass by reference in methods between objects. This means inheritance and containment is not used for class relationships.
    b. Following Java Coding Conventions
    c. Code Readability (as discussed in class)
    d. Uses packages (default package does not count)
    e. Code compiles and executes
    f. Adequately tested (with at least 3 unique test cases covering boundary conditions)
    g. Class diagram is provided (UML usage is not required)