CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 12, 13
Lab 06 – Summer 2020

Objective:

After today's lab, the students should be able:

- To describe objects and classes, and use classes to model objects
- To access an object's data and methods using the object member access operator (.)
- To define private data fields with appropriate getter and setter methods
- To encapsulate data fields to make classes easy to maintain
- To use the keyword **this** to refer to the calling object itself

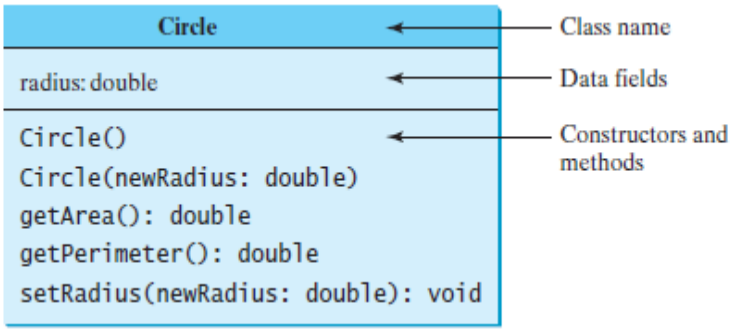
```
class Circle {
/** The radius of this circle */
  double radius = 1;
/** Construct a circle object */
  Circle() {
  }
/** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }
/** Return the area of this circle */
  double getArea() {
    return radius * radius * Math.PI;
  }
/** Return the perimeter of this circle */
  double getPerimeter() {
    return 2 * radius * Math.PI;
  }
/** Set new radius for this circle */
  double setRadius(double newRadius) {
    radius = newRadius;
  }
}
```

Data field

Constructors

Method

UML Class Diagram



Task – 1

(*The **Rectangle** class*) Following the example of the **Circle** class in Section 9.2, design a class named **Rectangle** to represent a rectangle. The class contains:

■ Two **double** data fields named **width** and **height** that specify the width and height of the rectangle. The default values are **1** for both **width** and **height**.

■ A no-arg constructor that creates a default rectangle.

■ A constructor that creates a rectangle with the specified **width** and **height**.

■ A method named **getArea()** that returns the area of this rectangle.

■ A method named **getPerimeter()** that returns the perimeter.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two **Rectangle** objects—one with width **4** and height **40** and the other with width **3.5** and height **35.9**. Display the width, height, area, and perimeter of each rectangle in this order.

Task – 2

(*The **Stock** class*) Following the example of the **Circle** class in Section 9.2, design a class named **Stock** that contains:

■ A string data field named **symbol** for the stock's symbol.

■ A string data field named **name** for the stock's name.

■ A **double** data field named **previousClosingPrice** that stores the stock price for the previous day.

■ A **double** data field named **currentPrice** that stores the stock price for the current time.

■ A constructor that creates a stock with the specified symbol and name.

■ A method named **getChangePercent()** that returns the percentage changed from **previousClosingPrice** to **currentPrice**.

Draw the UML diagram for the class and then implement the class. Write a test program that creates a **Stock** object with the stock symbol **ORCL**, the name **Oracle Corporation**, and the previous closing price of **34.5**. Set a new current price to **34.35** and display the price-change percentage.

Task – 3

(*The **Account** class*) Design a class named **Account** that contains:

■ A private **int** data field named **id** for the account (default **0**).

■ A private **double** data field named **balance** for the account (default **0**).

■ A private **double** data field named **annualInterestRate** that stores the current interest rate (default **0**). Assume all accounts have the same interest rate.

■ A private **Date** data field named **dateCreated** that stores the date when the account was created.

■ A no-arg constructor that creates a default account.

■ A constructor that creates an account with the specified id and initial balance.

■ The accessor and mutator methods for **id**, **balance**, and **annualInterestRate**.

■ The accessor method for **dateCreated**.

■ A method named **getMonthlyInterestRate()** that returns the monthly interest rate.

■ A method named **getMonthlyInterest()** that returns the monthly interest.

■ A method named **withdraw** that withdraws a specified amount from the account.

■ A method named **deposit** that deposits a specified amount to the account.

Draw the UML diagram for the class and then implement the class. (*Hint*: The method **getMonthlyInterest()** is to return monthly interest, not the interest rate. Monthly interest is **balance * monthlyInterestRate**. **monthlyInterestRate** is **annualInterestRate / 12**. Note that **annualInterestRate** is a percentage, e.g., like 4.5%. You need to divide it by 100.)

Write a test program that creates an **Account** object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the **withdraw** method to withdraw $2,500, use the **deposit** method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

Task – 4

(*The **Fan** class*) Design a class named **Fan** to represent a fan. The class contains:

■ Three constants named **SLOW**, **MEDIUM**, and **FAST** with the values **1**, **2**, and **3** to denote the fan speed.

■ A private **int** data field named **speed** that specifies the speed of the fan (the default is **SLOW**).

■ A private **boolean** data field named **on** that specifies whether the fan is on (the default is **false**).

■ A private **double** data field named **radius** that specifies the radius of the fan (the default is **5**).

■ A string data field named **color** that specifies the color of the fan (the default is **blue**).

■ The accessor and mutator methods for all four data fields.

■ A no-arg constructor that creates a default fan.

■ A method named **toString()** that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string "fan is off" in one combined string.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two **Fan** objects. Assign maximum speed, radius **10**, color **yellow**, and turn it on to the first object. Assign medium speed, radius **5**, color **blue**, and turn it off to the second object. Display the objects by invoking their **toString** method.