WIA1002 DATA STRUCTURE
WEEK 1 & 2
OBJECT-ORIENTED PROGRAMMING

Demonstrating an example:
(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).
- A no-arg constructor that creates a default fan.
- The accessor and mutator methods for all four data fields.
- 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.

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 toStringmethod.

1. The MyPoint Class

Design a class named MyPoint to represent a point with x- and y-coordinates. The class contains:

- The data fields x and y that represent the coordinates with getter methods.
- A no-arg constructor that creates a point (0, 0).
- A constructor that constructs a point with specified coordinates.
- A method named distance that returns the distance from this point to a specified point of the MyPoint type.
- A method named distance that returns the distance from this point to another point with specified x- and y-coordinates.
- A static method named distance that returns the distance from two MyPoint objects.

Write a test program that creates the two points (0, 0) and (10, 30.5) and displays the distance between them.

2. ATM machine

(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 0.1). Assume that 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.

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, for example 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.

(Modified implementation)
Use the Account class created to simulate an ATM machine. Create 10 accounts in an array with id 0, 1, …, 9, and an initial balance of $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter choice 1 for viewing the current balance, 2 for withdrawing, 3 for depositing money, and 4 for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will not stop.

3. The Course class

- A data field named courseName.
- An array named students.
- A data field named numberOfStudents.
- A constructor that takes in the courseName.
- An accessor for course name.
- A method addStudent to add student, automatically increase the array size if there is no room to add more students.
- A method to drop student.

- Accessor for student list, getStudents() method.
- Accessor for number of students.
- Add a new method named clear() that removes all students from the course.

4. Geometry: the Circle2D class

Define the Circle2D class that contains:

- Two double data fields named x and y that specify the center of the circle with getter methods.
- A data field radius with a getter method.
- A no-arg constructor that creates a default circle with (0,0) for (x,y) and 1 for radius.
- A constructor that creates a circle with the specified x, y, and radius.
- A method getArea() that returns the area of the circle.
- A method getPerimeter() that returns the perimeter of the circle.
- A method contains(double x, double y) that returns true if the specified point (x, y) is inside this circle (see Figure 10.21a).
- A method contains(Circle2D circle) that returns true if the specified circle is inside this circle (see Figure 10.21b).
- A method overlaps(Circle2D circle) that returns true if the specified circle overlaps with this circle (see Figure 10.21c)

Write a test program that creates a Circle2D object c1 (new Circle2D(2, 2, 5.5)), displays its area and perimeter, and displays the result of c1.contains(3, 3), c1.contains(new Circle2D(4, 5, 10.5)), and c1.overlaps(new Circle2D(3, 5, 2.3)).