COMP20220 Programming II (Conversion)

Practical 6

Q1 Download the `Circle` and `TestCircle` classes from Moodle. Use these classes as a starting point for this question.

To begin, update the `Circle` class such that it contains:

- A private `double` data field named `radius` (default 1).
- Private `double` data fields named `x` and `y` to represent the coordinates of the center point of a circle (default x = 0, y = 0).
- A private static `int` data field `numberOfObjects` to represent the number of objects created.
- A public no-arg constructor that creates a default circle.
- A public constructor that creates a circle with the specified radius and x and y coordinates.
- The public getter (accessor) methods for `radius`, `x` and `y`.
- The public setter (mutator) method for `radius`.
- A public static method to return the number of objects created.
- A public method named `move(double x, double y)` that moves this circle to the specified x and y coordinates (i.e. changes the center point of the circle).
- A public method named `getArea()` that returns the area.
- A public method named `getPerimeter()` that returns the perimeter.
- A public method named `getDistance(Circle another)` that returns the distance between the center point of this circle and another circle.
- A public `toString()` method that returns a string representation of a circle (i.e. the radius and center point).

Refer to the UML diagram for this class shown in Figure 1 and implement the class.

Then write a test program (**note: this program is available on Moodle**) that creates a `Circle` object with radius 5 and center point (2, -1). Display the circle (using the `toString()` method). Move the circle to point (3, 4) and change its radius to 10. Display the circle again.

Create a second `Circle` object with default values for the data fields. Display the circle. Display the distance between the center points of the first and second circles. Finally, display the number of `Circle` objects created.

UML Class Diagram

Circle	
−radius: double	The radius of this circle (default: 1.0)
−x: double	The x coordinate of this circle's center point (default: 0.0)
−y: double	The y coordinate of this circle's center point (default: 0.0)
−numberOfObjects: int	The number of circle objects created
+Circle()	Constructs a default circle object
+Circle(radius: double, x: double, y: double)	Constructs a circle object with the specified radius and x and y coordinates
+getRadius(): double	Returns the radius of this circle
+getX(): double	Returns the x coordinate of this circle's center point
+getY(): double	Returns the y coordinate of this circle's center point
+getNumberOfObjects(): int	Returns the number of circle objects created
+setRadius(radius: double): void	Sets a new radius for this circle
+move(x: double, y: double): void	Moves the center point of this circle
+getArea(): double	Returns the area of this circle
+getPerimeter(): double	Returns the perimeter of this circle
+getDistance(another: Circle): double	Returns the distance between the center points of this circle and another circle
+toString(): String	Returns a string representation of this circle

Note:

+ indicates a public modifier

− indicates a private modifier

underline indicates a static data field or method

Figure 1: The UML diagram for class `Circle`.

Q2 Design a class named `Stock` that contains:

- A private string data field named `symbol` for a stock's symbol.
- A private string data field named `name` for a stock's name.
- A private `double` data field named `previousClosingPrice` that stores the stock price for the previous day (default 0).
- A private `double` data field named `currentPrice` that stores the current stock price (default 0).
- A public constructor that creates a stock with the specified symbol and name.
- The public getter (accessor) methods for `symbol` and `name`.
- The public getter (accessor) and setter (mutator) methods for `previousClosingPrice` and `currentPrice`.
- A public method named `getChangePercent()` that returns the price-change percentage from `previousClosingPrice` to `currentPrice`.
- A public `toString()` method that returns a string representation of a stock (i.e. the symbol, name, previous closing price, and current price).

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. Set the previous closing price to 100.00. Set the current price to 90.00 and display the stock and the price-change percentage.

Q3 Design a class named `Account` that contains:

- A private `int` data field named `id` for an account (default 0).
- A private `double` data field named `balance` for an 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 (i.e. this data field should be static).
- A private `Date` data field named `dateCreated` that stores the date when an account was created.
- A public no-arg constructor that creates a default account.
- A public constructor that creates an account with the specified id and balance.
- The public getter (accessor) and setter (mutator) methods for `id`, `balance`, and `annualInterestRate`.
- The public getter (accessor) method for `dateCreated`.
- A public method named `getMonthlyInterest()` that returns the monthly interest amount (i.e. `balance` * `annualInterestRate` / 12).
- A public method named `withdraw` that withdraws a specified amount from an account.
- A public method named `deposit` that deposits a specified amount to an account.
- A public `toString()` method that returns a string representation of an account (id, balance, and date created).

Hint: see the Java API for details on the `java.util.Date` class. For example, `Date d = new Date()` creates a new `Date` object initialised to represent the date and time at which it was created. The statement `System.out.println(d)` displays the date.

Draw the UML diagram for the class and then implement the class.

Write a test program that creates an `Account` object with an account id of 1,122, a balance of $20,000, and set an annual interest rate of 0.045 (i.e. 4.5%). Use the withdraw method to withdraw $2,500 and use the deposit method to deposit $3,000. Display the balance, the monthly interest amount, and the date when the account was created.