EX1

(*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 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.

Draw the UML diagram for the class 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 `annualInterestRate` is a percentage, for example 4.5%. You need to divide it by 100.)

EX2

(*Game: ATM machine*) Use the `Account` class created in Programming Exercise 9.7 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 money, `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.

```
Enter an id: 4 Enter

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1 Enter
The balance is 100.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 2 Enter
Enter an amount to withdraw: 3 Enter

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1 Enter
The balance is 97.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 3 Enter
Enter an amount to deposit: 10 Enter
```

EX3)

(*The* `Course` *class*) Revise the `Course` class as follows:

- Revise the `getStudents()` method to return an array whose length is the same as the number of students in the course. (*Hint*: create a new array and copy students to it.)
- The array size is fixed in Listing 10.6. Revise the `addStudent` method to automatically increase the array size if there is no room to add more students. This is done by creating a new larger array and copying the contents of the current array to it.
- Implement the `dropStudent` method.
- Add a new method named `clear()` that removes all students from the course.

Write a test program that creates a course, adds three students, removes one, and displays the students in the course.

EX4)

Build Queue data structure with array

- isEmpty()
- peek()
- enqueue()
- dequeue()

EX5)

(*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.

Draw the UML diagram for the class then implement the class. Write a test program that creates the two points (`0`, `0`) and (`10`, `30.5`) and displays the distance between them.

EX6)

Rectangle2D

Build a Triangle2D class with p1, p2 and p3 all type of MyPoint class we built previously. Getters and setters.

- A no-arg constructor that creates a default triangle with the points (`0`, `0`), (`1`, `1`), and (`2`, `5`)
- A constructor that creates a triangle with the specified points.
- A method `getArea()` that returns the area of the triangle.
- A method `getPerimeter()` that returns the perimeter of the triangle.

(*Large prime numbers*) Write a program that finds five prime numbers larger than `Long.MAX_VALUE`.

(*Mersenne prime*) A prime number is called a *Mersenne prime* if it can be written in the form $2^p - 1$ for some positive integer p. Write a program that finds all Mersenne primes with $p \leq 100$ and displays the output as shown below. (*Hint*: You have to use `BigInteger` to store the number because it is too big to be stored in `long`. Your program may take several hours to run.)

```
   p          2^p - 1
----------------------

   2               3
   3               7
   5              31
```

(*Calculator*) Revise Listing 7.9, Calculator.java, to accept an expression as a string in which the operands and operator are separated by zero or more spaces. For example, `3+4` and `3 + 4` are acceptable expressions. Here is a sample run: