***10.1** (*The* `Time` *class*) Design a class named `Time`. The class contains:

- The data fields `hour`, `minute`, and `second` that represent a time.
- A no-arg constructor that creates a `Time` object for the current time. (The values of the data fields will represent the current time.)
- A constructor that constructs a `Time` object with a specified elapsed time since midnight, January 1, 1970, in milliseconds. (The values of the data fields will represent this time.)
- A constructor that constructs a `Time` object with the specified hour, minute, and second.
- Three getter methods for the data fields `hour`, `minute`, and `second`, respectively.
- A method named `setTime(long elapseTime)` that sets a new time for the object using the elapsed time. For example, if the elapsed time is `555550000` milliseconds, the hour is `10`, the minute is `19`, and the second is `10`.

Draw the UML diagram for the class and then implement the class. Write a test program that creates two `Time` objects (using `new Time()` and `new Time(555550000)`) and displays their hour, minute, and second in the format hour:minute:second.

(*Hint*: The first two constructors will extract the hour, minute, and second from the elapsed time. For the no-arg constructor, the current time can be obtained using `System.currentTimeMillis()`, as shown in Listing 2.7, ShowCurrentTime.java.)

10.3 (*The* `MyInteger` *class*) Design a class named `MyInteger`. The class contains:

- An `int` data field named `value` that stores the `int` value represented by this object.
- A constructor that creates a `MyInteger` object for the specified `int` value.
- A getter method that returns the `int` value.
- The methods `isEven()`, `isOdd()`, and `isPrime()` that return `true` if the value in this object is even, odd, or prime, respectively.
- The static methods `isEven(int)`, `isOdd(int)`, and `isPrime(int)` that return `true` if the specified value is even, odd, or prime, respectively.
- The static methods `isEven(MyInteger)`, `isOdd(MyInteger)`, and `isPrime(MyInteger)` that return `true` if the specified value is even, odd, or prime, respectively.
- The methods `equals(int)` and `equals(MyInteger)` that return `true` if the value in this object is equal to the specified value.
- A static method `parseInt(char[])` that converts an array of numeric characters to an `int` value.
- A static method `parseInt(String)` that converts a string into an `int` value.

Draw the UML diagram for the class and then implement the class. Write a client program that tests all methods in the class.

10.4 (*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.

Draw the UML diagram for the class and 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.

***10.5** (*Displaying the prime factors*) Write a program that prompts the user to enter a positive integer and displays all its smallest factors in decreasing order. For example, if the integer is `120`, the smallest factors are displayed as `5`, `3`, `2`, `2`, `2`. Use the `StackOfIntegers` class to store the factors (e.g., `2`, `2`, `2`, `3`, `5`) and retrieve and display them in reverse order.

***10.6** (*Displaying the prime numbers*) Write a program that displays all the prime numbers less than `120` in decreasing order. Use the `StackOfIntegers` class to store the prime numbers (e.g., `2`, `3`, `5`, `...`) and retrieve and display them in reverse order.