Worksheet Two — Object-Oriented Programming

The code for this worksheet can be found under the module source code repository under the directory `worksheet-two`; it is an IntelliJ project. For each exercise there is a corresponding IntelliJ *module* on the repository. **Please note**: only some of the exercise modules have populated code but you should use the module layout for your answers.

(+) indicated a *slightly* more challenging exercise.

1. Write a program that prints a lottery combination, picking six distinct numbers between 1 and 49. To pick six distinct numbers, start with an array list filled with `1..49`. Pick a random index and remove the element. Repeat six times. Print the result in sorted order. Write appropriate tests for your completed code using the provided method outlines.

2. The following table lists the freezing and boiling points of several substances (Fahrenheit):

Substance	Freezing Point	Boiling Point
Ethyl Alcohol	-173	172
Oxygen	-362	-306
Water	32	212

   Design a class that stores a temperature in a temperature field and has the appropriate accessor and mutator methods for the field. In addition to appropriate constructors, the class should have the following methods:

   isEthylFreezing — This method should return the boolean value `true` if the temperature stored in the temperature field is at or below the freezing point of ethyl alcohol. Otherwise, the method should return `false`.

   isEthylBoiling — This method should return the boolean value `true` if the temperature stored in the temperature field is at or above the boiling point of ethyl alcohol. Otherwise, the method should return `false`.

   isOxygenFreezing — This method should return the boolean value `true` if the temperature stored in the temperature field is at or below the freezing point of oxygen. Otherwise, the method should return `false`.

   isOxygenBoiling — This method should return the boolean value `true` if the temperature stored in temperature field is at or above the boiling point of oxygen. Otherwise, the method should return `false`.

   isWaterFreezing — This method should return the boolean value `true` if the temperature stored in the temperature field is at or below the freezing point of water. Otherwise, the method should return `false`.

   isWaterBoiling — This method should return the boolean value `true` if the temperature stored in the temperature field is at or above the boiling point of water. Otherwise, the method should return `false`.

   Write a program that demonstrate the class. The program should ask the user to enter a temperature, and then display a list of the substance that will freeze at that temperature and those that will boil at that temperature. For example, if the temperature is `-20` the class should report that water will freeze and oxygen will boil at that temperature.

   Sample output:

```
Enter a temperature: 48
Oxygen will boil.
```

3. (+) Extend the previous exercise.

- In each of the freezing or boiling methods, check that the method parameters are valid. Write a unit test to validate the behaviour you implemented.
- There is a consistent pattern of elements determining if the element will freeze or boil. While the class performs a specific behaviour, it isn't set up to adapt to new elements. Refactor the existing code to use a `Enum`.
- For each of the freezing and boiling methods we wrote a positive test to validate if it met a condition. Write the negative test for each method validating the false behaviour.

4. Consider the `nextInt` method of the `Scanner` class. Is it an *accessor* or a *mutator*? Why? What about the `nextInt` method of the `Random` class?

5. Can you ever have a mutator method return something other than `void`?
   Can you ever have an accessor method return `void`?
   Give examples where possible.

6. Why can't you implement a Java method that swaps the contents of two `int` variables?
   - Illustrate your answer with an appropriate example.
   - Write a method that swaps the contents of two `IntHolder` objects.
     (Look up this rather obscure class in the API documentation.)
   - Can you swap the contents of two `Integer` objects?

7. (+) Implement an *immutable* class `Point` that describes a point in a plane. Provide a constructor to set it to a specific point, a no-arg constructor to set it to the origin, and methods `getX`, `getY`, `translate`, and `scale`.

   The `translate` method moves the point by a given amount in `x`-direction and `y`-direction.
   The `scale` method scales both coordinates by a given factor.

   Implement these methods so that they return new points with the results.
   Write appropriate unit tests for these methods.

   Example usage of the class:

   ```
   Point p = new Point(3, 4).translate(1, 3).scale(0.5);
   ```

   should set `p` to a point with coordinates `(2, 3.5)`.

8. Repeat the preceding exercise, but now make `translate` and `scale` into *mutators*.

9. Add Javadoc comments to both versions of the `Point` class from the preceding exercises.

10. In the preceding exercises, providing the constructors and getter methods of the `Point` class was rather repetitive. Most IDEs have shortcuts for writing the boilerplate code. What does the IntelliJ IDE offer?

11. (+) Implement a class `Car` that models a car traveling along the `x`-axis, consuming fuel as it moves.
    Provide methods to drive by a given number of kilometres, to add a given number of litres to the fuel tank, and to get the current distance from the origin and fuel level.

    Specify the fuel efficiency (in km/litres) in the constructor. Should this be an immutable class? Why or why not?

12. (+) In the `RandomNumbers` class in the repository, provide two `static` methods called `randomElement` that get a random element from an *array* or *array list* of integers. (Return `zero` if the array or array list is *empty*.

    Why couldnt you make these methods into instance methods of `int[]` or `ArrayList<Integer>`?