- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.
- Failure to export properly will result in your work not getting marked.

To submit:

- Export your project to a JAR file, with source code.
- Name your JAR file ID_Week12_Q1.jar. For example, 6623110021_Week12_Q1.jar
- Submit the JAR file on MyCourseville.

You are writing code for sorting numbers in array but the sorting condition is special.

- **Copy all Java files into your Eclipse project**.

File **Sort.java** is given. Also, **Pair.java**, is given. A Pair stores value and frequency.

- You can make modifications to both **Sort.java** and **Pair.java**
- **You can create a new Pair data by calling new Pair(v,f):**
  - **v is a value**
  - **f is the value's frequency**
- `Pair[] p = new Pair[size];` can be used to create an array of Pair. (But you will also need to create a new pair within each array slot).

In Sort.java, write method **public static int[] sortFrequency(int[] x)**:

- This method receives array x, which contains only numbers 1 to 10 but the numbers can be duplicated.
- The method returns an array that puts the numbers with higher frequency first (no duplicated data will be shown in this result array).
- If the frequency of 2 data are equal, put smaller values first.

Example:

We want to sort the following array, x, according to frequency:

6	7	6	4	3	1	2	3	1	3	1	3	2	3	5

From the array, we can see that:

- Value 1 has frequency = 3
- Value 2 has frequency = 2
- Value 3 has frequency = 5
- Value 4 has frequency = 1
- Value 5 has frequency = 1
- Value 6 has frequency = 2

- Value 7 has frequency = 1

Therefore the result array from **sortFrequency(int[] x)**, which arrange data according to frequency, and does not have duplicate value is:

3	1	2	6	4	5	7

Scoring Criteria (10 marks total)

- Using fast sorting algorithm (2 marks)

JUnit tests are in **TestSort.java**

- testSort1 (2 marks)
- testSort2 (2 marks)
- testSort3 (1 mark)
- testSort4 (1 mark)
- testSort5 (2 marks)