Excercise No.1
(Do you remember Java Language?)

Q1

Initialize int typed array with random numbers. Then implement a class called Sort.java that sorts given numbers by normal ascending order. Note that, all algorithm should be implemented in the main method.

An example of main method in Sort.java

```
public static void main(String[] args) {

    int[] array = {10,8,2,4,6,7,0};

    ..............

}
```

An examle of the output in case of (10,8,2,4,6,7,0) numbers)

```
%java Sort
0
2
4
6
7
8
10
```

Hint

- array.length gives the number that array contains.

Q2

Implement the following extension to the Sort class defined in Q1.

- Define sort method and use it.
- Define print method that displays contents of the array from the beginning.
- Invoke the above methods in main method and output the same result as Q1. Note that we can define the array typed int in the main method.

Interfaces of sort and print method as follows.

sort method

```
public void sort(int[] array);
```

print method

```
public void print(int[] array);
```

An examle of the output in case of (10,8,2,4,6,7,0) numbers)

```
%java Sort2
0
2
4
6
7
8
10
```

Hints

- Both of sort and print are NOT static methods.
- Creating an instance of Sort2 is required to invoke each method.

Q3

Implement "Hello.java" that reads strings from standard input, then display it by concatenating it with the string "Hello". If it read nothing, this program displays "Hello World". Note that, you can implement everything in the main method or define several methods based on the their rolls.

Execution example

```
%java Hello
Please input your name -> Taro
Hello Taro

%java Hello
Please input your name -> Jiro
Hello Jiro


%java Hello
Please input your name ->
Hello World
```

Hint

The following pieces of code shows the example to read one line from standard input (java.io package should be imported).

```
BufferedReader reader =
            new BufferedReader(new InputStreamReader(System.in));
String line = reader.readLine();
```

Exception can be handled by a try-catch clause, or just thorws the caller.

Q4

Implement StrCompare class that has a method called comapre. This method reads two kinds of string from standard input, then returns true if the former string is included in the latter string, otherwise it returns false. Not that, you cannot use indexOf prepared by String class to enforce this class. In contrast, you can use toCharArray() to translate from String to an array of characters.

Definision of compare method

```
public boolean compare(String first, String second);
```

Execution example

```
%java StrCompare
Input First String -> test123
Input Second String -> 123
judge = true

%java StrCompare
Input First String -> test123
Input Second String -> 1234
judge = false

%java StrCompare
Input First String -> test123
Input Second String -> test123
judge = true

%java StrCompare
Input First String -> test123
Input Second String -> st12
judge = true
```

Hint

- Translate string to characters by using toCharArray(), then compares every character.

Q5

We found several hierarchies in nature. For example the relation among Human, Dog, Bird, Mammal and Animal is:

- Human is a kind of Mammal
- Dog is a kind of Mammal
- Mammal is a kind of Animal
- Brid is a kind of Animal

In addition, each layer in this hierarchy has the following features:

- Animals finally die.
- Mammal brings a child.
- Bird brins an egg.
- Mammal moves on foot.
- Bird moves by flying.
- Human moves on two legs
- Dog moves on four legs

We now represent four features such as "what kind in animals", "the means to move", "whether it will die or not" and "how it brings a child" as following methods.

- What kind in animals - String getKind()
- Means to move - String getMove()
- Whether it will die or not - boolean willDie()
- how it brings a child - String getHowChild()

Let's design classes of these animals by using inheritance of Object-oriented programming. In addition, each method should return the return values as follows

getKind

- "Human" in case of Human.
- "Dog" in case of Dog.
- "Bird" in case of Bird.

getMove

- "Four legs walking" in case of Human
- "Two legs walking" in case of Dog
- "Fly" in case of Bird.

willDie

- always true

getHowChild

- "as Egg" if Bird
- "as Child" if Mammal

Implement these classes above. Also, Let's implement a class called Feature that read a string from standard input. If the string is "inu", this class needs to display the feature of Dog. Likewise, if the string is "human" or "bird", it should display the feature of Human or Bird respectively. In addition, the Feature class must have a check method that display each feature. If the string read is out of "human", "bird" and "inu", the program will exit with string "Error".

An interface of check method

```
public void check(Animal animal);
```

Execution example

```
%java Feature
Input : inu
Kind:  Dog
Move:  Four legs walking
Child: as Child
Die  : true

%java Feature
Input : bird
Kind:  Bird
Move:  Fly
Child: as Egg
Die  : true

%java Feature
```

```
Input : human
Kind:  Human
Move:  Two legs walking
Child: as Child
Die  : true

%java Feature
Input : neko
Error
```