Assignments for Week-9
abstraction class & interface

`Converter` class is as follow (For Q1 ~ 2) :

```
import java.util.Scanner;
abstract class Converter {
  abstract protected double convert(double src);
  abstract protected String getSrcString();
  abstract protected String getDestString();
  protected double ratio;

  public void run() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Convert "+getSrcString()+" to "+getDestString());
  System.out.print("Enter "+getSrcString()+" >>> ");
  double val = scanner.nextDouble();
  double res = convert(val);
  System.out.print(val+" "+getSrcString()+" is converted to "+res+"
"+getDestString());
  scanner.close();
  }
}
```

1. Create `Won2Dollar` class which inherits the `Converter` class. (main() method and the execution result are as follows) :

   ```
   public static void main(String[] args) {
     Won2Dollar toDollar = new Won2Dollar(1200.0);
     toDollar.run();
   }
   ```

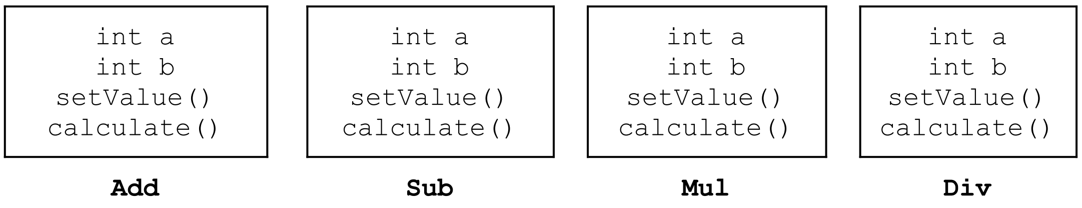
   ```
   Convert KRW to USD
   Enter KRW >>> 24000
   24000.0 KRW is converted to 20.0 USD
   ```

2. Create `Km2Mile` class which inherits the `Converter` class. (main() method and the execution result are as follows) :

   ```
   public static void main(String[] args) {
     Km2Mile toMile = new Km2Mile(1.6);
     toMile.run();
   }
   ```

   ```
   Convert km to Mile
   Enter km >>> 30
   30.0 km is converted to 18.75 mile
   ```

3. Mike wants to create 4 classes (`Add, Sub, Mul, Div`) which has 3 fields & methods :
    - a, b field of int data type for operands
    - `void setValue(int a, int b)` : Save operands in the object.
    - `int calculate()` : Calculate according to the operand (Add. Sub, Mul, Div) and returns the result.



After the lecture about “abstract class”, Mike found that he can declare the abstract class `Calc` and create classes which inherits class `Calc`.

Define abstract class `Calc` & make a program which calculates the operands as follows. Create an object according to the operator (`+` create object of `Add` class, – create object of `Sub` class and so on)

```
Enter 2 operands & operator >>> 5 + 7
12
```

(Input ‘5’, ‘+’, ‘7’ means it needs to create `Add` class since it has ‘+’ operand)

4. The interface to represent the Shape is as follow :

```
interface Shape {
  final double PI = 3.14;
  void draw();
  double getArea();
  default public void redraw() {
    System.out.print("-- Redraw : ");
    draw();
  }
}
```

Refer the following `main()` method and result to write `Circle` class which implements `Shape` interface.

```
public static void main(String[] args) {
      Shape donut = new Circle(10); // Radius is 10
      donut.redraw();
      System.out.println("Area : "+ donut.getArea());
}
```

```
-- Redraw : Circle with radius 10.0
Area : 314.0
```

5. Use `Shape` interface of Q4, Refer the following `main()` method and result to write `Rect` class which implements `Shape` interface.

```
public static void main(String[] args) {
      Shape [] lit = new Shape[2];
      list[0] = new Circle(10);      // Circle with Radius 10
      list[1] = new Rect(10, 40);    // 10x40 Rectangle

      for(int i=0;i<list.length;i++) list[i].redraw();
      for(int i=0;i<list.length;i++)
            System.out.println("Area : "+ list[i].getArea());
}
```

```
-- Redraw : Circle with radius 10.0
-- Redraw : Rectangle with size 10.0x40.0
Area : 314.0
Area : 400.0
```