Java Programming
Lab FP 1

Functional Programming Basics

1. Lab objectives

This initial lab is designed to help get started with the basics of using functions as first class objects.

2. Lab Setup

A VM with Java and Eclipse have been provided. These labs will be assuming you are using that configuration in your VM. VSCode is also available in the VM if you choose to use that.

You may use your own machine and IDE if you choose; however, there is no time available during the course to assist in trouble shooting any set-up or configuration issues you may have on your personal computer.

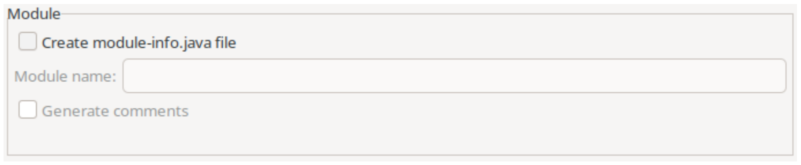
If you are having issues with the provided VM, there is technical support available as discussed in class.

3. Lab Solutions

Since the demos are based on the labs, if you are having problems getting started, refer to the demo code in the repository to give you some hints.

4. Getting Started

1. Open up a new Eclipse workspace and create a new Java project called `FP1`. Ensure that the option to to "`Create module-info.java file`" is *not* selected.



Module
☐ Create module-info.java file
Module name:
☐ Generate comments

2. Create a new package, the solution code uses `lab1` as the package name.

3. Create a class with a `main()` method. The solution uses the class name `Main`.

5. Create Some Functions

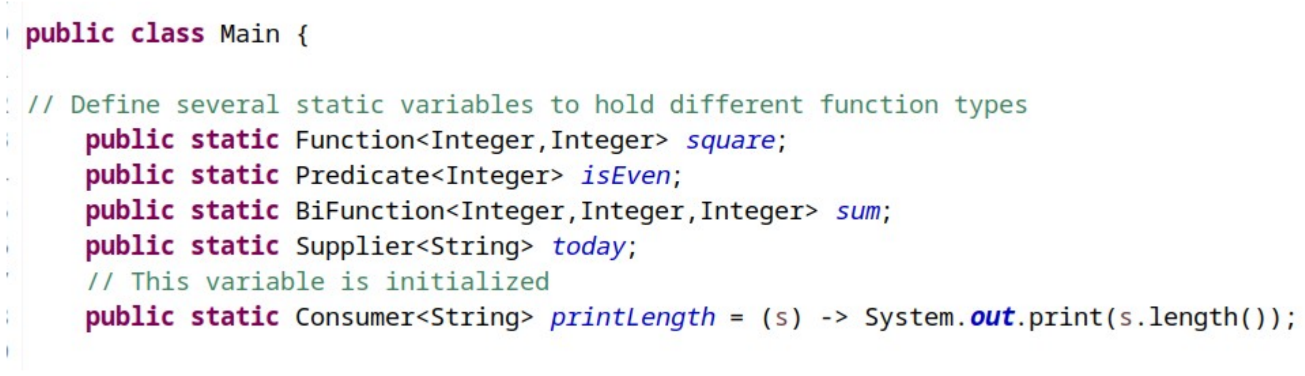
Remember that Java does not support global variables. We can use variables that hold functions in the same places we create variables that hold data. These are:

1. As Instance variables in a class definition
2. As local variables in a method body
3. As a static variable in a class definition.

In the demo and in the lab, only the last two are used. The static variables are in the Main class in the solutions but you can put them in any class you want.

Create five variables to hold functions of different types:

1. A variable called **square** that holds a function that takes and Integer argument and returns an integer
2. A variable called **isEven** that holds a predicate that takes an Integer argument.
3. A variable called **sum** that takes two Integers and arguments and returns an Integer.

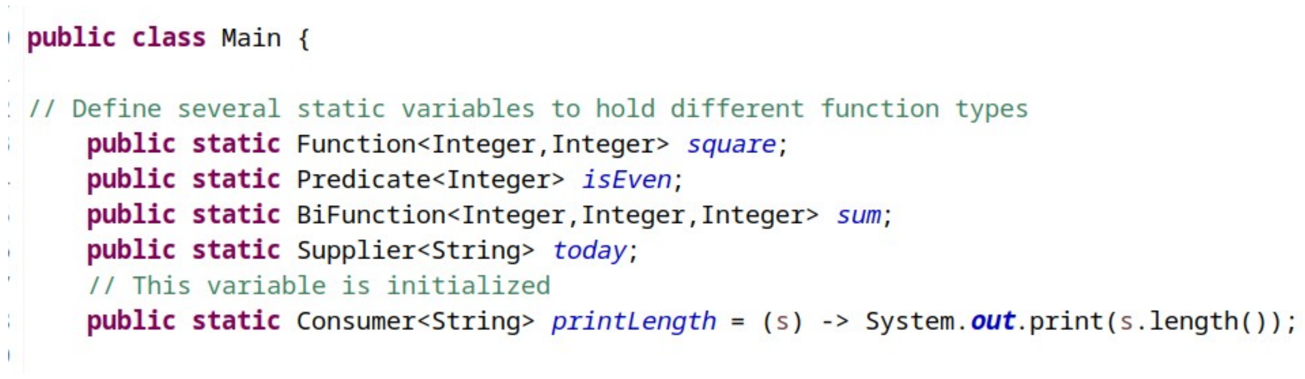


```
public class Main {

// Define several static variables to hold different function types
    public static Function<Integer,Integer> square;
    public static Predicate<Integer> isEven;
    public static BiFunction<Integer,Integer,Integer> sum;
    public static Supplier<String> today;
    // This variable is initialized
    public static Consumer<String> printLength = (s) -> System.out.print(s.length());
```

4. A variable called **today** that holds a Supplier that returns a String.
5. A variable called **printLength** that takes a string argument.

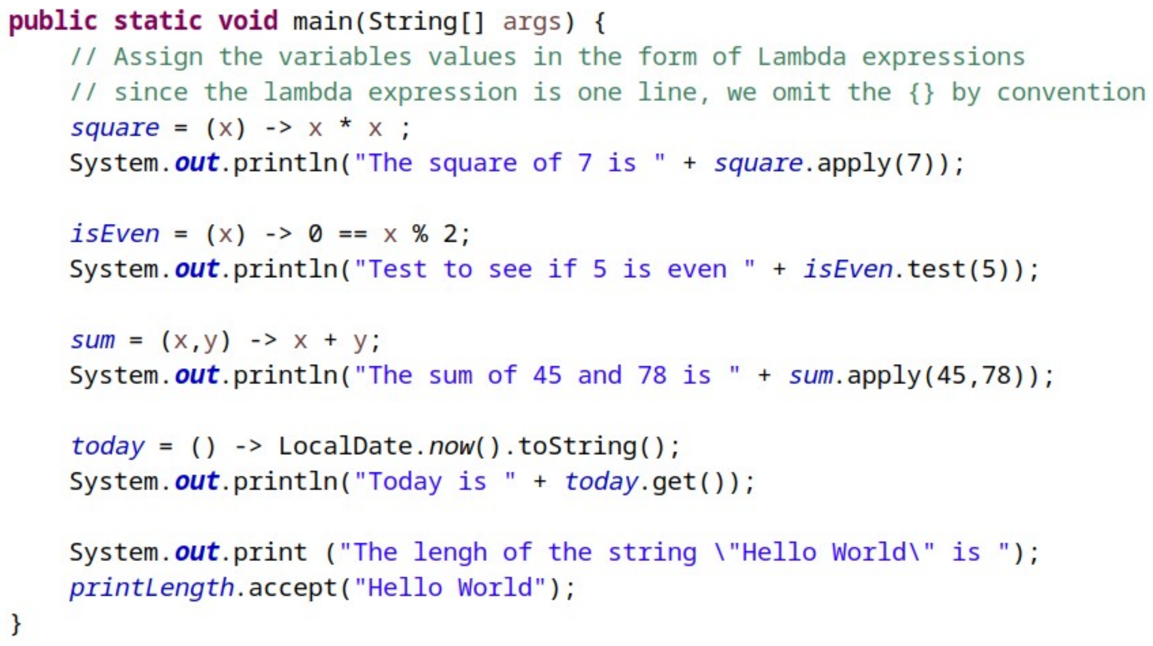
Initialize the **printLength** method with the function body

(s) -> System.out.print(s.length())



```java
public class Main {

// Define several static variables to hold different function types
    public static Function<Integer,Integer> square;
    public static Predicate<Integer> isEven;
    public static BiFunction<Integer,Integer,Integer> sum;
    public static Supplier<String> today;
    // This variable is initialized
    public static Consumer<String> printLength = (s) -> System.out.print(s.length());
```

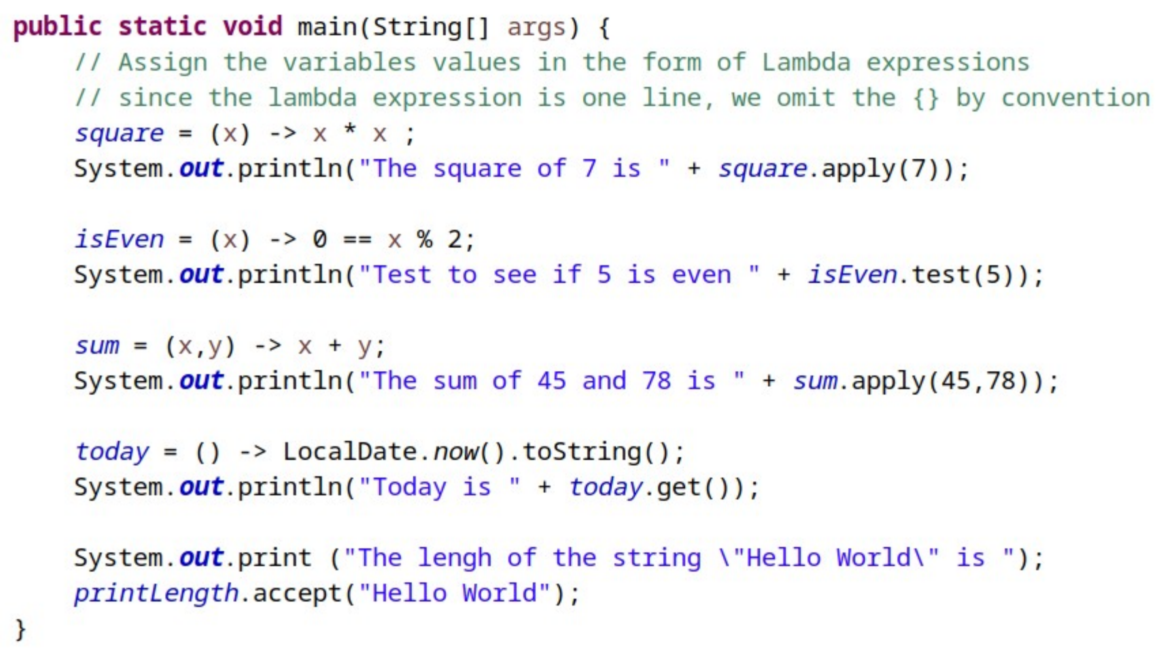
In the body of the main method, assign the Lambda functions to the variables as shown. Then use each function with some test data and the appropriate apply(), test(), get() or accept() method. Print out the result of executing each function as shown below:



```java
public static void main(String[] args) {
    // Assign the variables values in the form of Lambda expressions
    // since the lambda expression is one line, we omit the {} by convention
    square = (x) -> x * x ;
    System.out.println("The square of 7 is " + square.apply(7));

    isEven = (x) -> 0 == x % 2;
    System.out.println("Test to see if 5 is even " + isEven.test(5));

    sum = (x,y) -> x + y;
    System.out.println("The sum of 45 and 78 is " + sum.apply(45,78));

    today = () -> LocalDate.now().toString();
    System.out.println("Today is " + today.get());

    System.out.print ("The lengh of the string \"Hello World\" is ");
    printLength.accept("Hello World");
}
```

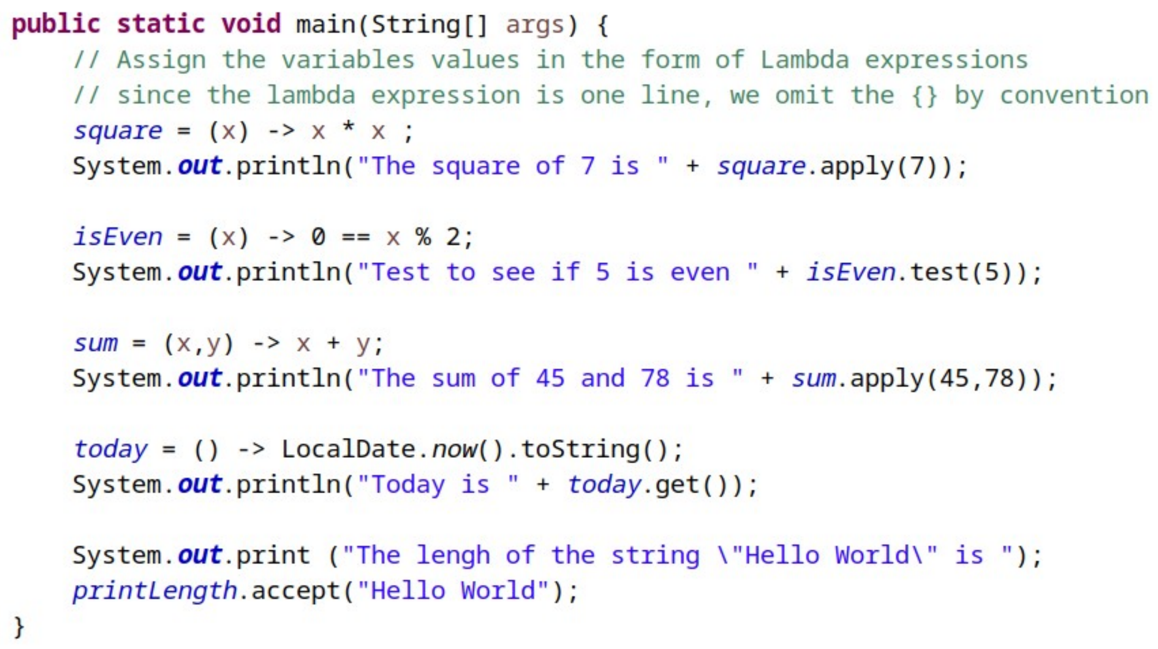
6. Assign function values

Create a new local function variable which is of type **Function<Integer,Integer>** called **other**.



```java
public static void main(String[] args) {
    // Assign the variables values in the form of Lambda expressions
    // since the lambda expression is one line, we omit the {} by convention
    square = (x) -> x * x ;
    System.out.println("The square of 7 is " + square.apply(7));

    isEven = (x) -> 0 == x % 2;
    System.out.println("Test to see if 5 is even " + isEven.test(5));

    sum = (x,y) -> x + y;
    System.out.println("The sum of 45 and 78 is " + sum.apply(45,78));

    today = () -> LocalDate.now().toString();
    System.out.println("Today is " + today.get());

    System.out.print ("The lengh of the string \"Hello World\" is ");
    printLength.accept("Hello World");
}
```

Assign the variable **square** to **other**. Print out the address of each of the variables as shown below. Notice that both variables contain exactly the same function body located at the address that is printed out. Assign a new Lambda function to **other** and print out the address.



```
public static void main(String[] args) {
    // Assign the variables values in the form of Lambda expressions
    // since the lambda expression is one line, we omit the {} by convention
    square = (x) -> x * x ;
    System.out.println("The square of 7 is " + square.apply(7));

    isEven = (x) -> 0 == x % 2;
    System.out.println("Test to see if 5 is even " + isEven.test(5));

    sum = (x,y) -> x + y;
    System.out.println("The sum of 45 and 78 is " + sum.apply(45,78));

    today = () -> LocalDate.now().toString();
    System.out.println("Today is " + today.get());

    System.out.print ("The lengh of the string \"Hello World\" is ");
    printLength.accept("Hello World");
}
```

End of Lab