Functional Programming

Combining Implementations

Helper Methods provided by FI's

- Consumer
  - `andThen()`
- Function
  - `andThen()`
  - `compose()`
- Predicate
  - `and()`
  - `or()`
  - `negate()`

```
import java.util.function.*;

public class MyClass {

    public static void main(String[] args) {

        Consumer<String> greet1 = s -> System.out.print("Hello, " + s + "! ");

        Consumer<String> greet2 = s -> System.out.print("Bye, " + s + "! ");

        Consumer<String> greetCombined = greet1.andThen(greet2);

        greetCombined.accept("John");


        // another way

        System.out.println();

        greet1.andThen(greet2).accept("John");

    }

}
```

```
Hello, John! Bye, John!

Hello, John! Bye, John!
```

```
import java.util.function.*;

public class MyClass {
  public static void main(String[] args) {
     Function<Integer, Integer> square = n -> n*n;
     Function<Integer, Integer> triple = n -> 3*n;
     Function<Integer, Integer> f1 = square.andThen(triple);
     Function<Integer, Integer> f2 = square.compose(triple);

     System.out.println(f1.apply(5));
     System.out.println(f2.apply(5));
  }
}
```

$5 \times 5 = 25$

$3 \times 25 = 75$

```
75

225
```

$(3 \times n) \times (3 \times n) = (3 \times 5) * (3 \times 5) = 225$

```
import java.util.function.*;

public class MyClass {
  public static void main(String[] args) {
    Predicate<Integer> gt10 = n -> n > 10;
    Predicate<Integer> lt20 = n -> n < 20;
    Predicate<Integer> p1 = gt10.and(lt20);
    Predicate<Integer> p2 = gt10.or(lt20);
    Predicate<Integer> p3 = lt20.negate();

    System.out.println(p1.test(5) + " " + p2.test(5) + " " + p3.test(5));
  }
}
```

```
false true false
```