# Programming Paradigms Tutorials
# Imperative programming

## Imperative programming

Imperative programming is a paradigm of computer programming in which the program describes a sequence of steps that change the state of the computer. Unlike declarative programming, which describes "what" a program should accomplish, imperative programming explicitly tells the computer "how" to accomplish it.

## References

Scala supports immutable and mutable coding styles, so the Scala programmer is responsible for knowing the two different styles and following the design philosophy of the codebase.

### Mutable references

Let’s use the var keyword to assign the variable x to an integer value.

```
var x: Int = 6
```

x can be reassigned to other integers, so it’s a mutable reference.

```
x = 10 // this works!
```

Scala is strongly typed, so once a variable is typed as an integer, it cannot be assigned to values of other types.

```
x = "bob" // error: type mismatch
```

### Immutable references

Let’s use the val keyword to assign the variable location to a string value via an immutable reference.

```
val location: String = "airplane"
```

location cannot be reassigned to another string value because it’s an immutable reference.

```
location = "school" // error: reassignment to val
```

### String Pool in JVM

Here is a diagram which clearly explains how String Pool is maintained in java heap space and what happens when we use different ways to create Strings.

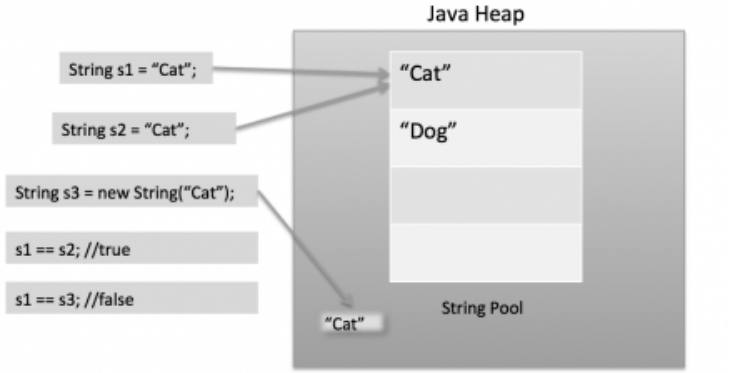


Fig. String Pool in Java

String Pool is possible only because String is immutable in Java and its implementation of String interning concept. String pool is also example of flyweight design pattern.

When we use double quotes to create a String, it first looks for String with the same value in the String pool, if found it just returns the reference else it creates a new String in the pool and then returns the reference. However, using new operator, we force String class to create a new String object in heap space. We can use intern() method to put it into the pool or refer to another String object from the string pool having the same value.

## Exercises

1. Write a method which return n-th Fibonacci number in
   a. Functional way (using recurrence)
   b. imperative way (using loop)

2. Write your own exception which will be thrown by method div when division by zero occurs.

3. What is printed and why by Java code:

```
public class IsEqual{
     static boolean isEqual1(int m, int n){
          return m == n;
     }
     static boolean isEqual2(Integer m, Integer n){
          return m == n;
     }

     public static void main(String[] args){
          System.out.println(isEqual1(250,250));
          System.out.println(isEqual2(250,250));
     }
}
```

4. What is printed and why by Java code:

```
public class StringEquasl{
     public static void main(String[] args){
          String s1 = "foo";
          String s2 = "foo";
          String s3 = new String("foo");
          System.out.println(s1 == s2);
          System.out.println(s1.equals(s2));
          System.out.println(s1 == s3);
          System.out.println(s1.equals(s3));
     }
}
```