Introduction to Programming

with Java, for Beginners

Method Frame
Call by Value
Stack
Global Variables

Frame for method calls
- Whenever a method is called, some memory is set aside to contain information related to the call
  - Example: parameter values, current statement being executed etc
  - This is called the frame for the method call

```java
int add ( int number1, int number2 ) {
    int sum = number1 + number2 ;
    return sum ;
}
```

- The frame is discarded when the method returns
  - This implies parameters and local variables (variables declared within method) are discarded
  - Hence, this explains the reasoning for scope rules.

Call by Value
- We call a method like this:
  ```java
  result = add( 3, 5 ) ;
  ```
  ```java
  int add ( int number1, int number2 ) {
      int sum = number1 + number2 ;
      return sum ;
  }
  ```

Call by value contd..
- Actual parameter values are copied into formally defined parameters

```java
int m = -1;
int result = absolute(m);
```  
  ```java
  int absolute(int n){
      if(n < 0){
          n = n * -1;
      }
      return n;
  }
  ```

- The value of m (i.e. -1) is copied to the formal parameters
- Changing n does not affect m
- The value of n is returned, but the variable n is thrown away
Methods calling other Methods

- boolean isLeapYear(int year){
  ....
}

void leapYearBet(int start, int end){
  int x = start;
  System.out.println("Leap years between "+ start + " and " + end + ":");
  while(x <= end){
    if(isLeapYear(x)){
      System.out.println(x);
    }
    x++;
  }
}

Stack

- Suppose a call of method m1 is being executed so that a frame for the call exists e.g. leapYears(2000, 2008)
- Suppose the method m1 while executing calls another method m2 e.g. leapYears method calls isLeapYear(x)
- A frame for m2 is created, & now at this point two frames exist
- If m2 calls another method m3, then a third frame is created
- The frames are destroyed in the order in which they were created
  - E.g. Call to m3 is first completed and hence discarded, then m2, and finally m1
  - Follows first-in-last-out principle and the section of memory that allocates memory during method calls is known as stack

Global Variables

- Variables that all methods can access
- These variables are placed in special section in memory
- They are alive until the program completes
- Declare them outside of all methods but within the class definition
  - E.g. static int global

Example of Global

public class Global {
  //Global Variable
  public static final double PI = 3.14;

  //Calculate area of a circle
  public static double area(double r){
    return r * r * PI;
  }

  //Calculate circumference of a circle
  public static double circum(double r){
    return 2 * PI * r;
  }
}