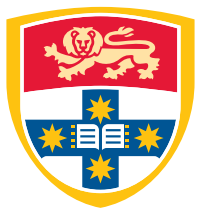
INFO1103: Introduction to Programming

School of Information Technologies, University of Sydney



Lecture 10: More Methods

Local variables and values

Methods: variables

Look at the variable num in the following

```
1  public class EvenNumber {
2      public static boolean isEven(int num) {
3          return (num % 2 == 0);
4      }
5
6      public static void main(String[] args) {
7          int num = 15;
8          boolean even = isEven(num);
9          System.out.println("even: " + even);
10     }
11 }
```

What is the output of this program?

Are all the variables called num the same?

Methods: Naming variables

The name of method arguments can be anything, it is not related to other parts of the program

```
1    public static boolean isEven(int myNumber) {
2        // ...
3    }
4    public static int findMax(int[] sailorMoonPowerUp) {
5        //...
6    }
```

e.g. argument `PrintStream` object can be named "ps" if it makes sense:

```
9    public static void printRow(PrintStream ps, int width) {
10       for (int i = 0; i < width; ++i) {
11           ps.print("*");
12       }
13       ps.println();
14   }
```

Local variables

Method arguments are new variables that exist only for the code block of the method {}. They are said to be local variables.

```
1    public static boolean isEven(int num) {
2        return (num % 2 == 0);
3    }
4    public static void main(String[] args) {
5        int num = 15;
6        boolean even = isEven(num);
7    }
```

What is the lifetime of num?

What about the value of num? is it stored in the same place?

What happens to my arguments?

Consider this piece of code:

```
1  public static int getSum(int x, int y) {
2      x += y;
3      return x;
4  }
```

and now suppose we call the `getSum` method from somewhere else, like this:

```
1          int x = 5;
2          int y = 7;
3          int s = getSum(x, y);
4          System.out.println("s: " + s);
5          System.out.println("x: " + x);
```

What is the value of x at the end?

Method call and local variables

When we call a method some values may be given as *arguments*.

When the method begins, it receives the *values* of those arguments.

Local variables are made to store those values, so they can be used. This is equivalent to an = operation.

```
1    public int getSum(int local_x, int local_y) { ... }
```

```
1    getSum( 23, 2 );
```

`public int` getSum(`int` local_x **= 23**, `int` local_y **= 2**) { ... }

```
1    int num = 2;
2    getSum( num, num*2 );
```

`public int` getSum(`int` local_x **= num**, `int` local_y **= num*2**) { ... }

A method call

Consider this example

```
1    public static void increment(int x) {
2        x++;
3    }
```

`main` method:

```
1    public static void main(String[] args) {
2        int p;
3        increment(p);
4        System.out.println(p);
5    }
```

What is the outcome?

Using static methods: Math

The Math class has many static methods

static double sqrt(double a)
Returns the correctly rounded positive square root of a double value.

`main` method:

```
1    public static void main(String[] args) {
2        double p = 64.0;
3        double answer = Math.sqrt(p);
4        System.out.println("sqrt(p) = " + answer);
5        System.out.println("sqrt(p) = " + Math.sqrt(p) );
6    }
```

What is happening with the return value from Math.sqrt(p) in each case?

Using static methods: Math

static double random()

Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

`main` method:

```
1  public static void main(String[] args) {
2      int i = 0;
3      while (i < 100) {
4        double random = Math.random();
5        if ( random < 0.1 ) {
6          System.out.print("winner ");
7        }
8        System.out.println("random value = " + random);
9        i = i + 1;
10     }
11 }
```

How many times will "winner" be printed?

Methods with Objects

String type revisted

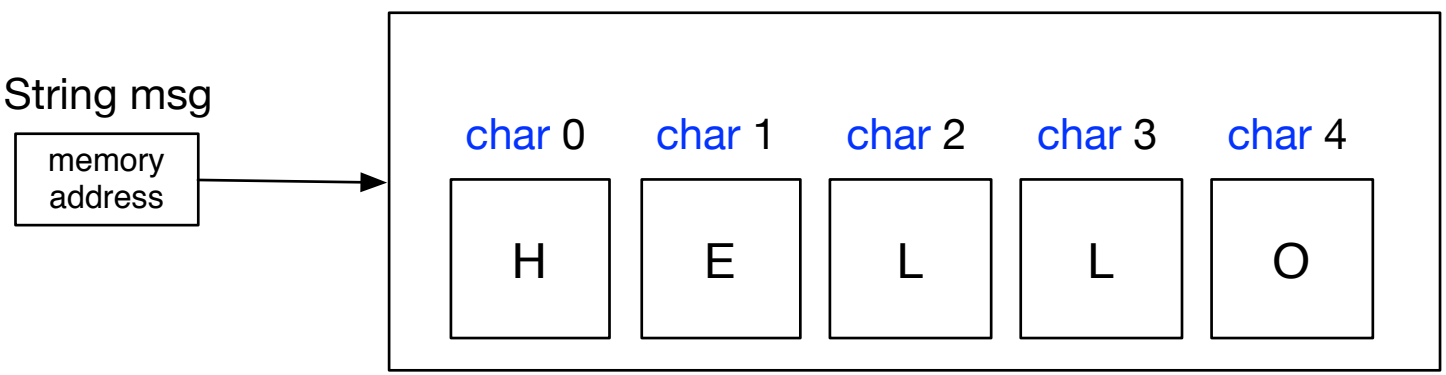
Recall the `String` type

`String` is a sequence of characters *stringed* together to represent text information.

`String` is an object, not a primitive type.

- it can have a few characters or several thousand.
- it has special operators to query or manipulate the text information.

```
1       String msg = "HELLO";
```



More `String` methods

`String` trim()

Returns a copy of the string, with leading and trailing whitespace omitted.

```
1 String msg = "   hello world  ";
2 System.out.println( msg.trim() );
3 System.out.println( msg );
```

`String` replace(`char` oldChar, `char` newChar)

Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.

```
1  String msg = "hello world";
2  System.out.println(  msg.replace('o', 'a') );
```

Method call chaining

Each of these methods will return a new String.

The new String that can also have a method called on it.

```
1 String msg = "     hello world     ";
2 System.out.println( msg.trim().toUpperCase() );
3 System.out.println( msg.toUpperCase().trim()  );
4 System.out.println( msg );
```

```
1 String msg = "    hello world      ";
2 System.out.println(
3    msg.trim().replace('o','a').replace('w','k') );
```

StringBuilder class

StringBuilder is a String that can be modified

String toString()
Returns a string representing the data in this sequence.

int reverse()
Causes this character sequence to be replaced by the reverse of the sequence.

```
1 String msg = "?siht daer uoy nac";
2 StringBuilder sb = new StringBuilder( msg );
3 System.out.println(sb.toString());
4 sb.reverse();
5 System.out.println(sb.toString());
```

The information stored within this StringBuilder is modified. There is no new copy returned. sb object has internal data.

StringBuilder class

The StringBuilder object sb stores its own information.

Another StringBuilder variable sb2 can *refer* to the object sb.
sb2 is assigned the reference value of sb using the = operator

```
1  String msg = "abc";
2  StringBuilder sb = new StringBuilder( msg );
3  StringBuilder sb2 = sb;
4
5  System.out.println(sb.toString());
6  System.out.println(sb2.toString());
7
8  sb.reverse();
9
10 System.out.println(sb.toString());
11 System.out.println(sb2.toString());
```

Variables sb and sb2 are referring to the same single object in memory.

StringBuilder class

What is the output of the following?

```
1  String msg = "POOL";
2  StringBuilder sb = new StringBuilder( msg );
3  StringBuilder sb2 = sb;
4
5  sb.reverse();
6
7  sb = new StringBuilder( msg );
8
9  System.out.println(sb.toString());
10 System.out.println(sb2.toString());
```

What does line 3 do?

What does line 7 do?

How many objects exist after line 7?

Methods with Objects like `StringBuilder`

StringBuilder is an example of how objects in Java behave with the assignment operator =

How does this affects method calls?

(Recall) Method calls and local variables

When we call a method some values may be given as *arguments*.

When the method begins, it receives the *values* of those arguments.

Local variables are made to store those values, so they can be used. This is equivalent to an = operation.

```
1    public static void addQuestionMark(StringBuilder local_sb) { ...
```

```
1    StringBuilder sb = new StringBuilder( "Soggy Waffles" );
2    addQuestionMark( sb );
```

`public int` addQuestionMark(StringBuilder local_sb **= sb**) { ... }

Methods with Objects like `StringBuilder` cont.

What is the output of the following?

```
1  public static void addQuestionMark(StringBuilder sb) {
2      sb.append('?');
3  }
4  public static void main(String[] args) {
5      StringBuilder sb = new StringBuilder( "Fish Milkshake" );
6      System.out.println(sb.toString());
7      addQuestionMark(sb);
8      System.out.println(sb.toString());
9  }
```

What value gets passed to method at line 1?

How many objects exist at each line of code?