Lecture 2

Object Oriented Programming

Value Returning Methods

1

Value returning methods

- In the last practical the methods created did not return any value to the calling section of code.
- This is limiting. It is often useful to return a value
- You have already used *value returning* methods, for example

```
int first = keyIn.nextInt();
```

- The method `nextInt()` returns an `int` value
- How do we create value returning methods?

2

Create methods that return values

- A ***method*** may return any *type* of data
- Syntax/General Form of method

```
accessSpecifier returnType methodName (parameter_list)
{
   statements;
}
```

- `returnType` specifies the *return type* of a method
- can return any type of data
  - E.g. `boolean, double, int, char…`
  - `String`

3

Example

- A method to return the smaller of two numbers:

```
public static int getSmallest(int no1, int no2)
{
   if (no1 < no2)
      return no1;
   else
      return no2;
}
```

Note the method has a return type, `int`

The keyword `return` is used to indicate the value that is to be returned to the calling code.

4

Example

- This method would be called by the following code in the `main` method:

```
      .....
      int smallest;
      smallest = getSmallest(number1, number2); // method call
      System.out.println("The smallest  number is: "+smallest);
   } // end main
   public static int getSmallest(int no1, int no2)
   {
      if (no1< no2)
         return no1;
      else
         return no2;
   }
} // end class
```


Example in full

```
import java.util.Scanner;
public class SmallestNumber
{
   public static void main(String [] args)
   {
      Scanner keyboardIn = new Scanner(System.in);
      int number1, number2, smallest;

      System.out.print("Please enter first number: ");
      number1 = keyboardIn.nextInt();
      System.out.print("Please enter second number: ");
      number2 = keyboardIn.nextInt();
      //method call - returned value assigned to smallest
      smallest = getSmallest(number1, number2);
      System.out.println("The smallest  number is: " + smallest );
   }//end main method

   //method returns the smaller of two int values
   public static int getSmallest(int no1, int no2)
   {
      if (no1 < no2)
         return no1;
      else
         return no2;
   }

}//end class
```

Returning values

- The *type* of the value a method returns must match with a variable that the value will be assigned to

```
int smallest ;
smallest = getSmallest(number1, number2);
  //method getSmallest() returns an int
```

- A value is returned to the calling section of code using the ***return*** statement.

7

Returning values

- When ***return*** statement is encountered the method returns immediately. No statements after it will be executed.
- The value associated with the ***return*** statement need not be a constant - it can be any valid expression
  - eg

```
return 10;

return length;

return x + y;

return x < y;
```

8

You try it

Given the following pseudocode:

GET mark
DETERMINE IF mark is a pass
PRINT result

Write a value returning method ***isPass()*** which takes one integer parameter and returns true if the parameter is 40 or greater.

9

Answer

```
public static boolean isPass(int mark)
{
    if (mark >= 40)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

10

Answer (alternative)

```
public static boolean isPass(int mark)
{
   return mark >= 40;
} // end isDivisible
```

this is a boolean expression

You try it

Write a method that takes appropriate parameter/s and calculates and returns the area of a circle.

```
public static double calcArea(double r)
{
          return 3.14159 * r * r;
} //end method
```

How would you call this method?

Write a method that takes a appropriate parameter/s and calculates and returns the area of a circle.

```
public static double calcArea(double r)
{
            return 3.14159 * r * r;
} //end method
```

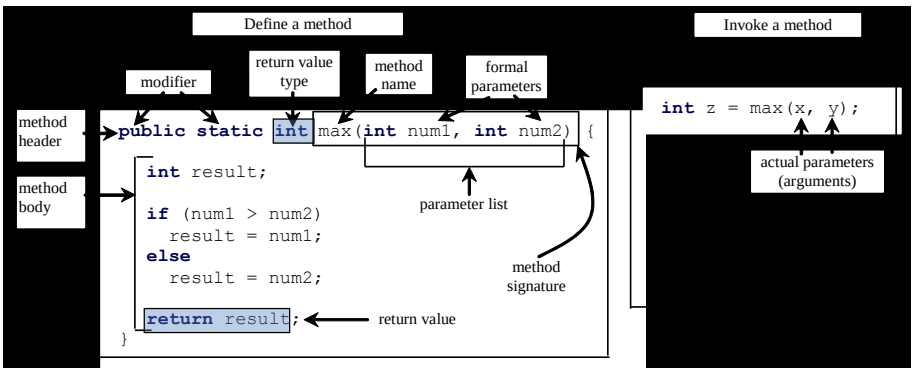
How would you call this method?

```
public static void main(String[] args)
{
         Scanner keyIn = new Scanner(System.in);
         double radius, area;

         System.out.print("\n\nEnter radius: ");
         radius = keyIn.nextDouble();
         //call the method
         area = calcArea(radius);

         System.out.print(" Area of circle with radius " + radius +" is " + area);

} //end main method
```

13

Methods - Summary

A method may return a value. The returnValueType is the data type of the value the method returns. If the method does not return a value, the returnValueType is the keyword void. For example, the returnValueType in the main method is void.



14

Calling Methods

The next example traces the calling of a method

The `max()` method and the Java code that calls it:

Testing the Max Method

```
public class TestMax
{
   /** main method */
   public static void main(String[] args)
   {
        int i = 5;
        int j = 2;
        int k = max(i, j);
        System.out.println("The maximum is " + k);

   } //end main method */

    /** Return the bigger of two numbers */
   public static int max(int num1, int num2)
   {
        int result;
        if (num1 > num2)
             result = num1;
         else
             result = num2;
         return result;
    }
} //end class
```

animation

Calling Methods, cont.

pass the value of i

pass the value of j

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

17

animation

Trace Method Invocation

i is now 5

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

18

animation

Trace Method Invocation

j is now 2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

19

animation

Trace Method Invocation

invoke max(i, j)

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

20

animation

Trace Method Invocation

invoke max(i, j)
Pass the value of i to num1
Pass the value of j to num2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

21

animation

Trace Method Invocation

declare variable result

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

22

animation

Trace Method Invocation

(num1 > num2) is true since num1 is 5 and num2 is 2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

23

animation

Trace Method Invocation

result is now 5

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

24

animation

Trace Method Invocation

return result, which is 5

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
pu  tatic int max(int num1, int num2) {
    esult;

    num1 > num2)
    esult = num1;
    se
    result = num2;

  return result;
}
```

25

animation

Trace Method Invocation

return max(i, j) and assign the return value to k

```
public static void main(Strin  ars) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

26

animation

Trace Method Invocation

Execute the print statement

```
public static void main(String[] ds) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

27