Faheem Akhtar Rajputt

Object Oriented Programming (JAVA)

Lecture 15

Nested Classes

- It is possible to define a class within another class, *nested classes*.
- Thus, if class B is defined within class A, then B does not exist independently of A
- An inner class is a non-static nested class.
- An inner class has access to all of the members of its enclosing class, but the reverse is not true.

Nested Classes - Example

```
// Demonstrate an inner class.
class Outer {
  int outer_x = 100;

  void test() {
    Inner inner = new Inner();
    inner.display();
  }

  // this is an inner class
  class Inner {
    void display() {
      System.out.println("display: outer_x = " + outer_x);
    }
  }
}
```

```
class InnerClassDemo {
  public static void main(String args[]) {
    Outer outer = new Outer();
    outer.test();
  }
}
```

Another Example

```
class Outer {
  int outer_x = 100;

  void test() {
    Inner inner = new Inner();
    inner.display();
  }

  // this is an innner class
  class Inner {
    int y = 10; // y is local to Inner
    void display() {
      System.out.println("display: outer_x = " +
outer_x);
    }
  }

  void showy() {
    System.out.println(y);  ←   // error, y not known here!
  }
}
```

This Program will not work

```
class InnerClassDemo {
  public static void main(String args[]) {
    Outer outer = new Outer();
    outer.test();
  }
}
```

Inner Class within any block scope

- it is possible to define inner classes within any block scope
- For example, you can define a nested class within the block defined by a method or even within the body of a for loop,

Example

```
// Define an inner class within a for loop.
class Outer {
  int outer_x = 100;

  void test() {
    for(int i=0; i<10; i++) {
      class Inner {
        void display() {
          System.out.println("display: outer_x = " + outer_x);
        }
      }
      Inner inner = new Inner();
      inner.display();
    }
  }
}
```

```
class InnerClassDemo {
  public static void main(String args[]) {
    Outer outer = new Outer();
    outer.test();
  }
}
```

OUTPUT:

```
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
```

Exploring String Class [1/3]

- The first thing to understand about strings is:
  - that every string you create is actually an object of type String
  - Even string constants are actually String objects

For eg.

```
System.out.println("This is a String, too");
```

Exploring String Class [2/3]

- The second thing to understand is:
- once a String object is created, its contents cannot be altered
  - If you need to change a string, you can always create a new one that contains the modifications
  - Java defines a peer class of String, called StringBuffer, which allows strings to be altered

String usage

```
String myString = "this is a test";
```

- Once created a string can be used in variety of ways.

```
System.out.println(myString);
```

- Concatenation in strings

```
String myString = "I" + " like " + "Java.";
```

- results in myString containing “I like Java.”

Example

OUTPUT
First String
Second String
First String and Second String

```
class StringDemo {
  public static void main(String args[]) {
    String strOb1 = "First String";
    String strOb2 = "Second String";
    String strOb3 = strOb1 + " and " + strOb2;
    System.out.println(strOb1);
    System.out.println(strOb2);
    System.out.println(strOb3);
    }
}
```

Useful methods of String class

- The String class contains several methods that you can use.
- equals( )
- length()
- charAt( )

General forms for all:

```
boolean equals(String object)

int length()

char charAt(int index)
```

Example

```
class StringDemo2 {
  public static void main(String args[]) {
    String strOb1 = "First String";
    String strOb2 = "Second String";
    String strOb3 = strOb1;

    System.out.println("Length of strOb1: " + strOb1.length());

    System.out.println("Char at index 3 in strOb1: ” +  strOb1.charAt(3));

    if(strOb1.equals(strOb2))
      System.out.println("strOb1 == strOb2");
    else
      System.out.println("strOb1 != strOb2");

    if(strOb1.equals(strOb3))
      System.out.println("strOb1 == strOb3");
    else
      System.out.println("strOb1 != strOb3");
  }
}
```

OUTPUT

```
Length of strOb1: 12
Char at index 3 in strOb1: s
strOb1 != strOb2
strOb1 == strOb3
```

Example

```
// Demonstrate String arrays.
class StringDemo3 {
  public static void main(String args[]) {
    String str[] = { "one", "two", "three" };

    for(int i=0; i<str.length; i++)
      System.out.println("str[" + i + "]: " +
                         str[i]);
  }
}
```

OUTPUT

```
str[0]: one
str[1]: two
str[2]: three
```

Varargs: Variable-Length Arguments

- A method that takes a variable number of arguments is called a variable-arity method, or simply a varargs method.
- A method that require variable number of arguments is not unusual.
  - Example: A method that interact with internet connection may ask for username, password, authentication file and protocol…
  - Another example is the printf( )

Old approach

- Previously variable-length arguments could be handled two ways:
  - First, if the maximum number of arguments was small and known, then you could create overloaded versions of the method
  - Second approach, used when number of arguments is unknown, then the array was passed to the method.

Example

```
// Use an array to pass a variable number of
// arguments to a method.
class PassArray {
  static void vaTest(int v[]) {
    System.out.print("Number of args: " + v.length + " Contents: ");

    for(int x : v)
      System.out.print(x + " ");
    System.out.println();
  }

  public static void main(String args[])
  {
    // Notice how an array must be created to
    // hold the arguments.
    int n1[] = { 10 };
    int n2[] = { 1, 2, 3 };
    int n3[] = { };

    vaTest(n1); // 1 arg
    vaTest(n2); // 3 args
    vaTest(n3); // no args
  }
}
```

OUTPUT

```
Number of args: 1 Contents: 10
Number of args: 3 Contents: 1 2 3
Number of args: 0 Contents:
```

Questions?


Sukkur
IBA
Thank you…