Inner Classes

CSCI 201L

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

- Inner Classes
Inner Class Rules

- An inner class, or nested class, is defined within the scope of another class
  - An inner class can be used just like a regular class, though it is typically only used by its outer class
  - An inner class can reference the data and methods defined in the outer class
  - An inner class can be defined with a visibility modifier (i.e. private)
  - If an inner class is defined as static, it can be accessed using the outer class name (not an instance)
    - The inner class will not be able to access nonstatic members of the outer class
    - To create an instance of a static inner class from a class other than the outer class, write
      ```java
      OuterClass.InnerClass ic = new OuterClass.InnerClass();
      ```
    - To create an instance of a nonstatic inner class from a class other than the outer class, write
      ```java
      OuterClass oc = new OuterClass();
      OuterClass.InnerClass ic = oc.new InnerClass();
      ```
public class Test {
    public Test() {
        Outer o = new Outer();
        Outer.Inner in = o.new Inner(3);
        System.out.println("num=" + in.getNum());
    }
    public static void main(String args[]) {
        Test t = new Test();
    }
}

class Outer {
    public Outer() {
        System.out.println("Outer constructor");
    }

    class Inner {
        private int num;
        public Inner(int num) {
            System.out.println("Inner constructor");
            this.num = num;
        }
        public int getNum() {
            return num;
        }
    }
}
public class Test {
    public Test() {
        Outer.Inner in = new Outer().new Inner(3);
        System.out.println("num=" + in.getNum());
    }
    public static void main(String args[]) {
        Test t = new Test();
    }
}

class Outer {
    public Outer() {
        System.out.println("Outer constructor");
    }
}

class Inner {
    private int num;
    public Inner(int num) {
        System.out.println("Inner constructor");
        this.num = num;
    }
    public int getNum() {
        return num;
    }
}
public class Test {
    public Test() {
        Outer.Inner in = new Outer.Inner(3);
        System.out.println("num="+in.getNum());
    }

    public static void main(String args[]) {
        Test t = new Test();
    }
}

class Outer {
    public Outer() {
        System.out.println("Outer constructor");
    }
}

class Inner {
    private int num;
    public Inner(int num) {
        System.out.println("Inner constructor");
        this.num = num;
    }

    public int getNum() {
        return num;
    }
}
Anonymous Inner Classes

- An anonymous inner class is an inner class without a name
  - It combines defining an inner class and creating an instance of the class
  - Anonymous inner classes always extend a superclass or implement an interface, but they cannot explicitly do so
  - Anonymous inner classes must implement all the abstract methods in the superclass or interface
  - Anonymous inner classes always use the default constructor from the superclass for instantiation

- Although anonymous inner classes can be used anytime, they are often used with event handling of GUI components
Anonymous Inner Class Example

```java
public class Test {
  public Test() {
    showValues(new Printer() {
      public void printNum() {
        System.out.println("num=" + 10);
      }
      public String getName() {
        return name;
      }
    });
  }
  public void showValues(Printer p) {
    p.printNum();
    System.out.println("name=" + p.getName());
  }
  public static void main(String args[]) {
    Test t = new Test();
  }
}

interface Printer {
  public static final int num = 10;
  public static final String name = "CSCI 201";
  public void printNum();
  public String getName();
}
```