HBO Graduaat Informatica Optie Programmeren

Java In Depth

Inner Classes

c v o   l e e r s t a d

v o l w a s s e n e n o n d e r w i j s

Inner Classes

- Inner classes nest within other classes.
- A normal class is a direct member of a package, a top-level class. Inner classes, come in four flavors:
  - Static member classes
  - Member classes
  - Local classes
  - Anonymous classes

Static Inner Classes

- Static member of a class (within top-level and other static member classes)
  - has access to all static methods of the top-level, class (including private members); the reverse is also true (Y.member)
  - outside of the containing class `X`, a static member class `Y` is named by combining the name of the outer class with the name of the inner class (`X.Y`)
- Can be instantiated
- Much like a regular top-level class, but declared locally for convenience

Static Inner Classes

- Example

```
public class StaticInnerTest {
  public static void main(String[] args){
    Enclosing enclosing =
      new Enclosing();
    enclosing.process(4);
    Enclosing.Enclosed enclosed =
      new Enclosing.Enclosed();
    enclosed.doSomething(”Peter");
  }
}
```

```
public class Enclosing {
  private static int encprivstat = 7;
  private int encpriv = 3;
  public void process(int i){
    for(int j=0;j<i;j++){
      System.out.print(".");
    }
    System.out.println("\n");
  }

  public static class Enclosed {
    private int priv = 5;
    public int pub = 20;
    private static int privstat = 10;
    public static int pubstat = 15;
    public void doSomething(String s) {
      for (int j=0;j<encprivstat;j++) {
        System.out.println(s);
      }
    }
  }

}
```

Member Classes

- Non-static member of a enclosing class
  - analoguous to a class field or class method
  - every ***member*** class is associated with an instance of the enclosing class
  - Instance specific: has access to any and all methods and members, even the parent's `this` reference.
  - Much like a regular top-level class, but declared locally for convenience

- No `static` members
- Exception: Only `static final` fields can be declared , no other `static` members are allowed

Member Classes

- Example

```java
public class MemberClassTest {
  public static void main(String[] args) {
    Enclosing2 enc2 = new Enclosing2();
    Enclosing2.Visitor visitor =
      enc2.visitor();
    while(visitor.hasMore()) {
      System.out.println(visitor.next());
    }
  }
}
```

```java
public class Enclosing2 {
  private static int encprivstat = 7;
  private int encpriv = 3;
  private int[] primes =
    new int[]{2, 3, 5, 7, 11, 13};
  public Visitor visitor() {
    return new Visitor();
  }

  public class Visitor {
    private int index = 0;
    public int next() {
      if(index < primes.length) {
        return primes[index++];
      } else {...}
    }
    public boolean hasMore() {
      return index <
        primes.length -  1;
    }
  }
}
```

- The use of local classes can sometimes eliminate the requirement to connect objects together via constructor parameters
- here: Visitor members have access to private *primes* variable

Local Classes

- Declared locally within a block of code
  - Visible only within that block, just as any other local method variable
  - However, in some cases, a local class can be defined closer to its point of use than would be possible with a member class, leading to improved code readability

Local Classes

- Example

```java
import java.util.Iterator;
public class LocalClassTest {
  public static void main(String[] args) {
  Enclosing3 enc3 = new Enclosing3();
  Iterator<Integer> visitor =
enc3.visit();
    while (visitor.hasNext()) {
      System.out.println(visitor.next());
    }
  }
}
```

This technique is frequently used in Java/Swing graphical applications

```java
public class Enclosing3 {
  private Integer[] primes =
    new Integer[]{2, 3, 5, 7, 11, 13};
  public Iterator<Integer> visit () {
    class Visitor
      implements Iterator<Integer> {

      private int index = 0;
      public Integer next() {
        if(index < primes.length) {
          return primes[index++];
        } else {...}
      }
      public boolean hasNext() {
        return index <
          primes.length -  1;
      }
      ...
    }
    return new Visitor();
  }

}
```

Anonymous Classes

- Local classes that have no name

```
import java.util.Iterator;
public class LocalClassTest {
  public static void main(String[] args) {
  Enclosing4 enc4 = new Enclosing4();
  Iterator<Integer> visitor =
enc4.visit();
    while (visitor.hasNext()) {
      System.out.println(visitor.next());
    }
  }
}
```

```
public class Enclosing4 {
  private Integer[] primes =
    new Integer[]{2, 3, 5, 7, 11, 13};
  public Iterator<Integer> visit() {
    return new Iterator<Integer>() {
      private int index = 0;
      public Integer next() {
        if(index < primes.length) {
          return primes[index++];
        } else {...}
      }
      public boolean hasNext(){
        return index < primes.length;
      }
      public void remove() {...}
    };
  }
}
```

This technique is frequently used in Java/Swing graphical applications

Questions ??