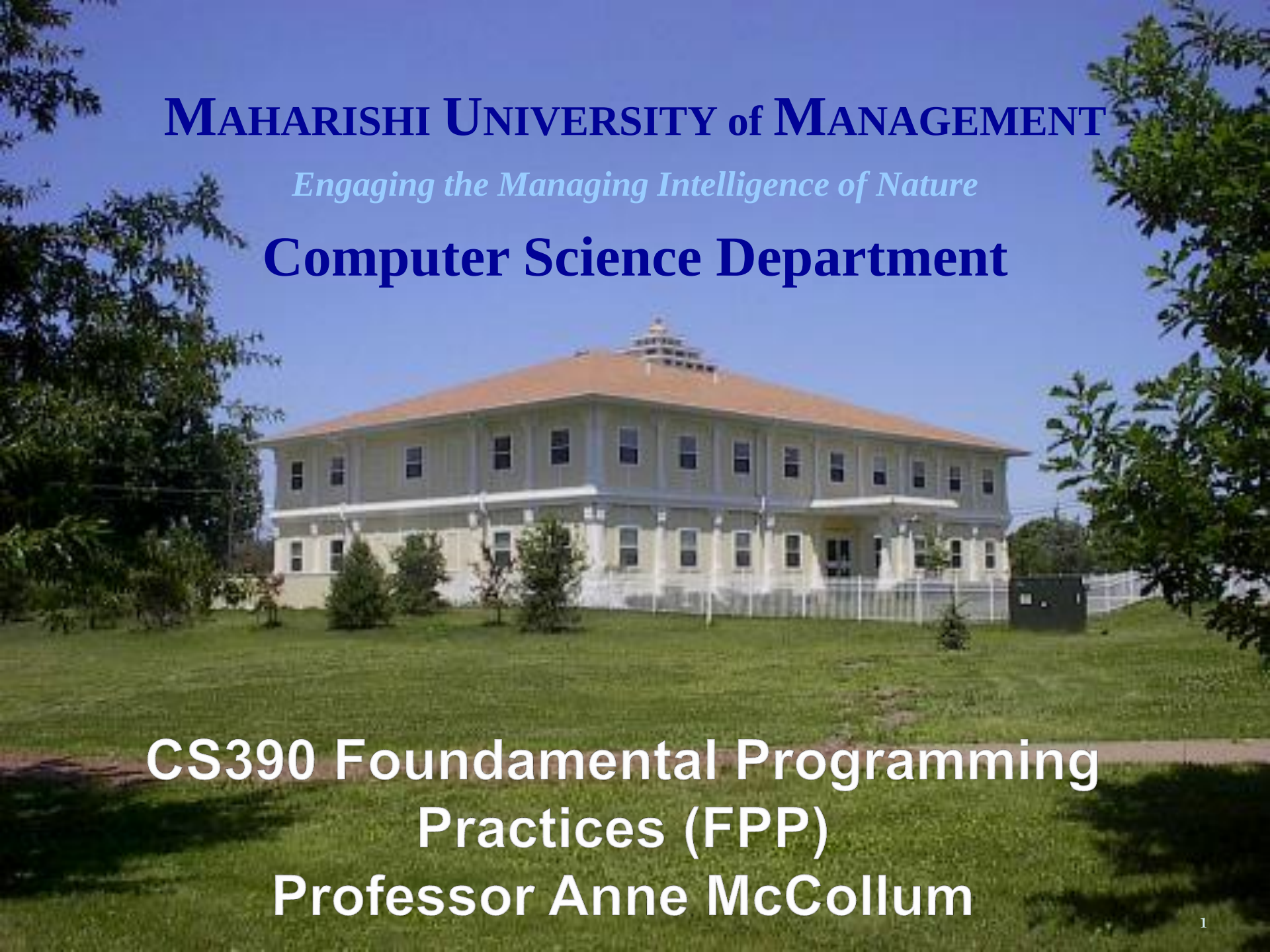

MAHARISHI UNIVERSITY of MANAGEMENT
Engaging the Managing Intelligence of Nature
Computer Science Department
CS390 Foundamental Programming Practices (FPP)
Professor Anne McCollum

Lecture 3b:
Nested Classes

Wholeness of the Lesson

Nested classes allow classes to play the roles of a variable: instance variable, static variable and local variable. This provides more expressive power to the Java language. Likewise, it is the hidden, unmanifest dynamics of consciousness that are responsible for the huge variety of expressions in the manifest world.

Outline of Topics

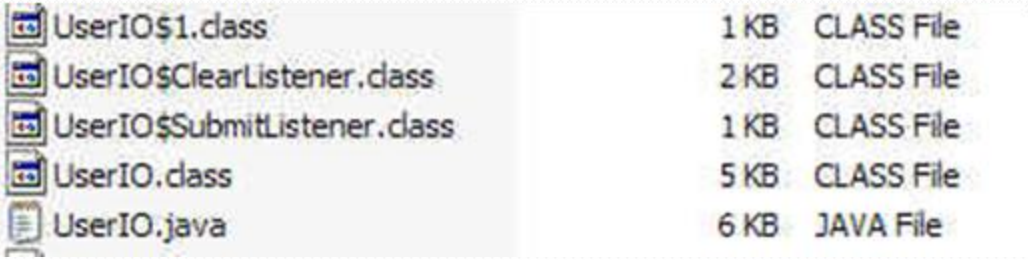
1. **Definitions of Nested and Inner Class**
2. Four Types of Nested Classes: Member, Static, Local, and Anonymous
3. Using Lambda Expressions in Place of Anonymous Inner Classes – MPP topic
   - Syntax of lambda expressions
   - Lambdas as implementers of functional interfaces
4. Comparator, Another Functional Interface
   - Example: Typical Implementation
   - Example: Implementation using an anonymous class
   - Example: Implementation using a lambda expression

Definition and Types of Nested Classes

- A class is a *nested class* if it is defined *inside* another class. (Note: This is different from having multiple classes defined in the same file.) A nested class is an *inner class* if it has access to all members (variables and methods) of its enclosing class (described below).
- Four kind of nested classes:
  - member
  - static
  - local
  - anonymous
- Of these, member, local, and anonymous are all called *inner* classes.
- An alternative, but equivalent, definition of *inner class* is "any non-static nested class"
- Sometimes in books you will see the term "static inner class" – this is simply loose language for static nested class.
- *See demos for nested classes in package* `lesson6_innerexamples`

Definition and Types of Nested Classes

- The compiler adds code to nested class definitions and to the enclosing class to support the features of nested classes. When you compile the `UserIO` class (which has two member inner classes), you will see the inner classes explicitly named in `.class` files, using a '$' in their names to distinguish them as nested classes.



Note: In `UserIO`, the `EventQueue.invokeLater` method defines an *anonymous* inner class – such a class is never given a name within the Java code. As the screen shot shows, the JVM handles this by naming such inner classes by number, starting with '1'.

- It is possible to make inner classes `private` and also `static` (see below) – these keywords cannot be used with ordinary classes.

Main Point

Classes are the fundamental concept in Java – programs are built from classes. With nested classes, Java makes it possible for this fundamental construct to play the roles of instance variable (member inner classes), static variable (static nested classes), and local variable (local inner classes). Likewise, in the unfoldment of creation, pure consciousness, pure intelligence assumes the role of creative intelligence. In all creation we find pure intelligence in the guise of individual expressions, individual existences, assuming diversified roles.

Outline of Topics

1. Definitions of Nested and Inner Class
2. **Four Types of Nested Classes: Member, Static, Local, and Anonymous**
3. Using Lambda Expressions in Place of Anonymous Inner Classes
   - Syntax of lambda expressions
   - Lambdas as implementers of functional interfaces
4. Comparator, Another Functional Interface
   - Example: Typical Implementation
   - Example: Implementation using an anonymous class
   - Example: Implementation using a lambda expression

Member Inner Class Example

```
public class InnerMemberExample {
    class StudentGrade {
        String studentLetterGrade;
        String studentId;

        public String toString() {
        return "Student with ID "+ studentId
                + " has grade " + studentLetterGrade;
        }
    }
    StudentGrade stGrade = new StudentGrade();
    {
        stGrade.studentLetterGrade = "A";
        stGrade.studentId = "123654";
    }
    void printGrade() {
        System.out.println(stGrade);
    }
    public static void main(String[] args) {
        new InnerMemberExample().printGrade();
    }
}
```

Static Class Example

```
public class InnerMemberExample {
    static class StudentGrade {
        String studentLetterGrade;
        String studentId;

        public String toString() {
        return "Student with ID "+ studentId
                + " has grade " + studentLetterGrade;
        }
    }
    StudentGrade stGrade = new StudentGrade();
    {
        stGrade.studentLetterGrade = "A";
        stGrade.studentId = "123654";
    }
    void printGrade() {
        System.out.println(stGrade);
    }
    public static void main(String[] args) {
        new InnerMemberExample().printGrade();
    }
}
```

Local Class Example

local to a method

```
public class LocalClassExample {
    void printGrade(String sGrade, String sGradeId) {
        class StudentGrade {
            String studentLetterGrade;
            String studentId;

            public String toString() {
            return "Student with ID "+ studentId
                + " has grade " + studentLetterGrade;
            }
        }
        StudentGrade stGrade = new StudentGrade();
        {
            stGrade.studentLetterGrade = sGrade;
            stGrade.studentId = sGradeId;
        }
        System.out.println(stGrade);
    }
    public static void main(String[] args) {
        new LocalClassExample().printGrade("A", "123654");
    }
}
```

Anonymous Class Example

```
public class AnonymousClassExample {
     interface IGrade {
          void printGradeInfo();
     }
     void printGrade(String sGrade, String sGradeId) {
          (new IGrade() {
               String studentLetterGrade = sGrade;
               String studentId = sGradeId;

               public String toString() {
                    return "Student with ID "+ studentId
                    + " has grade " + studentLetterGrade;
               }
               public void printGradeInfo() {
                    System.out.println(this);
               }
          }).printGradeInfo();
     }

     public static void main(String[] args) {
          new AnonymousClassExample().printGrade("A", "123654");
     }
}
```

Main Point

Inner classes – a special kind of nested class – have access to the private members of their enclosing class. The most commonly used kind of inner class is a *member* inner class. Likewise, when individual awareness is awake to its fully expanded, self-referral state, the memory of its ultimate nature becomes lively.

Summary

Java has four kinds of nested classes: member, static, local and anonymous

- *Member inner classes* are used as private support within a class, much as instance variables and private methods are used. They have full access to the instance variables and methods of the enclosing class.
- *Static nested classes* are top-level classes that are naturally associated with their enclosing class, but have no special access to the data or behavior of the enclosing class.
- *Local inner classes* are defined entirely within a method body; *anonymous inner classes* make it possible to define a class at the moment that an instance of the class is created. Both types of inner classes are accessible only within the local context in which they are defined, resulting in an extreme form of encapsulation.

Connecting the Parts of Knowledge With the Wholeness of Knowledge

Inner classes retain the memory of their "unbounded" context

1. A *nested class* is a class that is defined inside another class.
2. An *inner class* is a nested class that has full access to its context, its enclosing class.

---

3. **Transcendental Consciousness:** TC is the unbounded context for individual awareness.
4. **Wholeness moving within itself**: When individual awareness is permanently and fully established in its transcendental "context" – pure consciousness – every impulse of creation is seen to be an impulse of one's own awareness.