Intermediate Programming

Lecture 11 – UML

What is UML?

• Most people do not think in Java or other programming languages.
• Therefore, computer scientists have tried to devise more human-oriented ways to represent programs.
• These have included:
  – Flowcharts
  – Pseudocode
  – Structure diagrams
UML

• Unified Modeling Language (UML) is not a language in the normal sense; it is a graphical representation used in object-oriented programming (OOP) to show the contents of classes as well as the relationship between different classes in a program.

Class Diagrams

• A class diagram consists of three boxes, showing:
  – The name of the class (on top)
  – The data within the class (instance and static variables)
  – The actions (i.e., the methods) belonging to the class.
Notation for Visibility

• The properties and methods are marked to indicate their access:
  - indicates a private member
  + indicates a public member
  # indicates a protected member

• If there are members that are missing this is indicated by an ellipsis.

Example: UML Diagram for a Square
Inheritance Diagrams

• A inheritance diagram shows the classes and the hierarchy existing between base and derived classes.

Class Hierarchy in UML Notation

Person

Student
  Undergraduate
  Graduate

Employee
  Faculty
  Staff
Details in a UML Class Hierarchy

**Person**

- `name : String`

+ `setName(String newName): void`
+ `getName() : String`
+ `toString() : String`
+ `sameName(Person otherPerson) : boolean`

**Student**

- `studentNumber : int`

+ `set(String newName, int newStudentNumber): void`
+ `getStudentNumber() : int`
+ `setStudentNumber(int newStudentNumber): void`
+ `toString() : String`
+ `equals(Object otherObject) : boolean`

---

Example: Hourly and Salaried Employees

- Let's assume that we need to design a class for Employees that will include their names and dates of hire.
- We will include derived class for hourly employees and salaried employees, with their own properties and methods to handle the different manners in which they are paid.
Employee Class

• The Employee class will have two properties:
  – empName – consists of first name, middle initial and last name
  – hireDate – consists of month, day and year in numeric format.
• Both of these are properties are classes (Name and Date respectively).

Name.java

import java.util.Scanner;
public class Name
{
    // The properties: first name, middle initial
    // and last name
    private String first;
    private char initial;
    private String last;
// Default and conversion constructors
public Name()
{
    ...
}

public Name(String initFirst, char initInitial, String initLast) {
    ...
}

// read() – An input method
public void read() {
    ...
}

// Accessors
public String getFirst() {
    ...
}

public char getInitial() {
    ...
}

public String getLast() {
    return last;
}
// Mutators
public void setFirst(String newFirst) {
    //
}

public void setInitial(char newInitial) {
    //
}

public void setLast(String newLast) {
    //
}

// toString() – Converting the properties to a printable string
public String toString() {
    //
}

// equals() – Are the objects the same
public boolean equals(Name other) {
    //
}
UML Diagram for **Name**

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>- first: String</td>
</tr>
<tr>
<td>- initial : char</td>
</tr>
<tr>
<td>- last : String</td>
</tr>
<tr>
<td>+ Name(): void</td>
</tr>
<tr>
<td>+ Name(String aFirst, char anInitial, String aLast) :void</td>
</tr>
<tr>
<td>+ read() : void</td>
</tr>
<tr>
<td>+ getFirst(): String</td>
</tr>
<tr>
<td>+ getInitial() : char</td>
</tr>
<tr>
<td>+ getLast(): String</td>
</tr>
<tr>
<td>+ setFirst(String newFirst) : void</td>
</tr>
<tr>
<td>+ setInitial(char newInitial) : void</td>
</tr>
<tr>
<td>+ void setLast(String newLast) : void</td>
</tr>
<tr>
<td>+ toString(): String</td>
</tr>
<tr>
<td>+ equals(Name other) : boolean</td>
</tr>
</tbody>
</table>

---

**Date.java**

```java
import java.util.Scanner;
// A class to handle date; month, day and year
// all in numeric form
public class Date {
    // Our properties
    private int month;
    private int day;
    private int year;
```
// Default and Conversion constructors
public Date()
{
    ...
}

public Date(int initMonth, int initDay,
            int initYear) {
    ...
}

// read() – An input method
public void read() {
    ...
}

// Accessors
public int getMonth() {
    return month;
}

public int getDay() {
    return day;
}

public int getYear() {
    return year;
}

// Mutators
public void setMonth(int newMonth) {
    ...
}
```java
public void setDay(int newDay) {
    ...
}

public void setYear(int newYear) {
    ...
}

public String toString() {
    ...
}

public boolean equals(Date other) {
    ...
}
```

**UML Diagram for Date**

- int month : int
- int day : int
- int year : int

+ Date() : void
+ Date(int aMonth, int aDay, int aYear) : void
+ read() : void
+ getMonth() : int
+ getDay() : int
+ getYear() : int
+ setMonth(int newMonth) : void
+ setDay(int newDay) : void
+ setYear(int newYear) : void
+ toString() : String
+ equals(Date other) : boolean
Employee.java

import java.util.Scanner;

public class Employee
{
    // Properties: employee name and data of hire
    protected Name empName;
    protected Date hireDate;

    /// Default and conversion constructors
    public Employee()
    {
        ...
    }

    public Employee(Name aName, Date aDate)
    {
        ...
    }

    // read() – An input method
    public void read() {
        ...
    }

    // Accessors
    public Name getEmpName() {
        ...
    }

    public Date getHireDate() {
        ...
    }
}
// Mutators
public void setEmpName(Name newEmpName) {
    ...
}

public void setHireDate(Date newHireDate) {
    ...
}

public String toString() {
    ...
}

public boolean equals(Employee other) {
    ...
}
import java.util.Scanner;

public class HourlyEmployee extends Employee {
    // Two properties: hourly rate and hours worked
    private double hourlyRate;
    private double hoursWorked;

    // Default and conversion constructors
    public HourlyEmployee() {
        ...
    }

    public HourlyEmployee(Name initEmpName,
                           Date initHireDate, double initRate,
                           double initHours) {
        ...
    }

    //read() – An input method
    public void read() {
        ...
    }

    //Accessors
    public double getHourlyRate() {
        return hourlyRate;
    }
}
private double getHoursWorked() {
    ...
}

public void setHourlyRate(double newRate) {
    ...
}
private void setHoursWorked(double newHours) {
    ...
}

public String toString() {
    ...
}

public boolean equals(HourlyEmployee other) {
    ...
}

public double grossPay() {
    ...
}
}
UML Diagram for **HourlyEmployee**

```
HourlyEmployee
- hourlyRate : double
- hoursWorked : double

+ HourlyEmployee(): void
+ HourlyEmployee(Name anEmpName,
                    Date aHireDate, double anRate,
                    double anHours) : void
+ read() : void
+ getHourlyRate() : double
+ double getHoursWorked() : double
+ void setHourlyRate(double newRate) : void
- setHoursWorked(double newHours) : void
+ toString() : String
+ equals(HourlyEmployee other) : boolean
+ double grossPay()
```

**SalariedEmployee.java**

```java
import java.util.Scanner;

public class SalariedEmployee extends Employee
{
    // The only property
    private double annualSalary;

    //Default and conversion constructors
    public SalariedEmployee() {
        ...
    }
}
```
public SalariedEmployee(Name initEmpName, 
    Date initHireDate, double initSalary) {
    ...
}

// read() – An input method
public void read() {
    ...
}

public double getAnnualSalary() {
    ...
}

public void setAnnualSalary(double newRate) {
    ...
}

public String toString() {
    ...
}

public boolean equals(SalariedEmployee other) {
    ...
}

public double grossPay() {
    ...
}
UML Diagram for **HourlyEmployee**

```
SalariedEmployee
- annualSalary: double

+ SalariedEmployee() : void
+ SalariedEmployee(Name anEmpName,
    Date anHireDate, double anSalary) : void
+ read() : void
+ double getAnnualSalary() : double
+ void setAnnualSalary(double newRate) : void
+ toString() : String
+ equals(SalariedEmployee other) : boolean
+ grossPay() : double
```

The UML Class Diagram from BlueJ