Intermediate Programming

Lecture #10 – UML

What is UML?

- Most people do not think in Java or other programming lanaguges.
- 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.

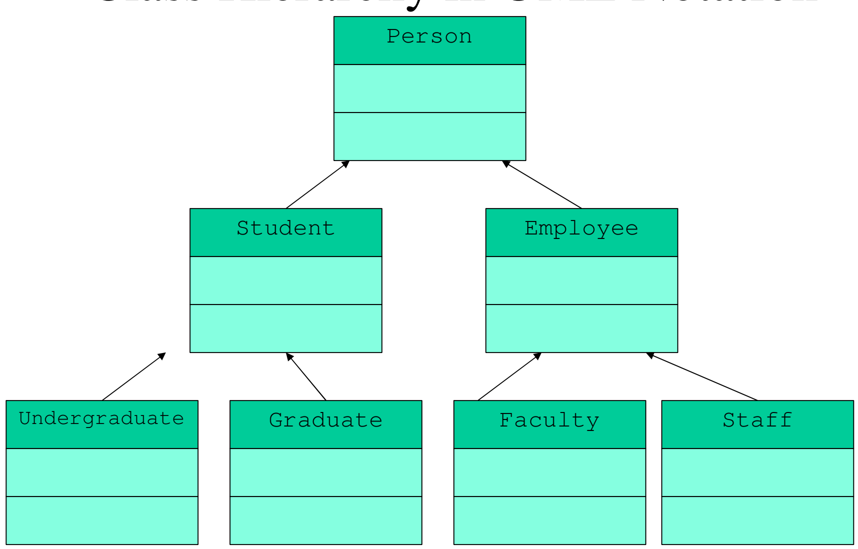
Example: UML Diagram for a Square

Square
- side : double - xCoordinate : double - yCoordinate : double
+resize(double newSide) : void + move (double newX, double newY): void # erase() : void …

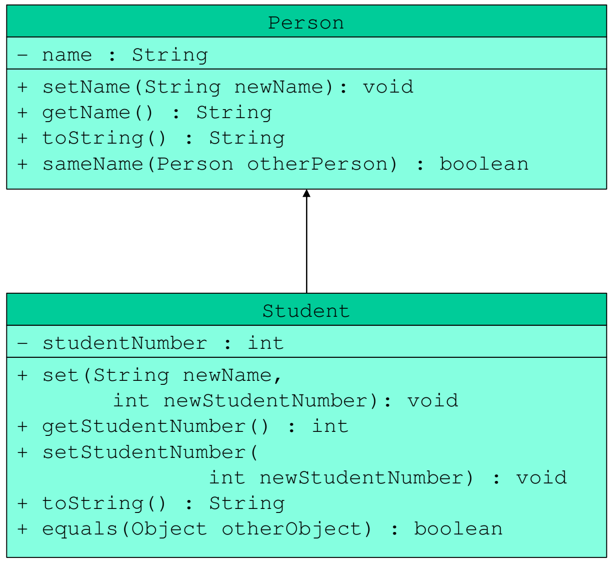
Inheritance Diagrams

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

Class Hierarchy in UML Notation



Details in a UML Class Hierarchy



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`**

```
Name
--------------------------------------------------
- first: String
- initial : char
- last : String
--------------------------------------------------
+ Name(): void
+ Name(String aFirst, char anInitial,
   String aLast) :void
+ read() : void
+ getFirst() : String
+ getInitial() : char
+ getLast() : String
+ setFirst(String newFirst) : void
+ setInitial(char newInitial) : void
+ void setLast(String newLast) : void
+ toString() : String
+ equals(Name other) : boolean
```

Date.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) {
    …
}
```

```
    public void setDay(int newDay) {
        …
    }

    public void setYear(int newYear) {
        …
    }

    public String toString() {
      …
    }

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

UML Diagram for **`Date`**

```
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(Name other) : boolean
```

Employee.java

```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()
    {
        …
    }
```

```java
    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) {
     …
    }
}
```

UML Diagram for **`Employee`**

```
Employee
-----------------------------------------------
# empName : Name
# hireDate : Date
-----------------------------------------------
+ Employee(): void
+ Employee(Name aName, Date aDate) : void
+ read() : void
+ getEmpName() : Name
+ getHireDate() : Date
+ setEmpName(Name newEmpName) : void
+ setHireDate(Date newHireDate) : void
+ toString() : String
+ equals(Name other) : boolean
```

HourlyEmployee.java

```
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;
    }
```

```java
    private double getHoursWorked() {
        …
    }

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

    public String toString() {
        …
    }
```

```java
    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`

```
import java.util.Scanner;

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

    //Default and conversion constructors
    public SalariedEmployee() {
        …
    }
```

```java
    public SalariedEmployee(Name initEmpName,
            Date initHireDate, double initSalary) {
        …
    }

    // read() - An input method
    public void read() {
        …
    }

     public double getAnnualSalary() {
        …
    }
```

```java
    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