Object Oriented Programming Lecture

Method Overriding & Overloading

School of Computing and Information Technology
Faculty of Engineering and Computing
University of Technology, Jamaica

Email: dwwhite@utech.edu.jm, taedwards@utech.edu.jm

Object Oriented Programming

Expected Outcome

At the end of this lecture the student should:

- understand method overriding
- understand method overloading
- differentiate between overriding and overloading
- implement overriding and overloading

Object Oriented Programming

Topics to be covered in this lecture:

- Methods
- The parts of a method
- Overriding
- Overloading
- Differentiating between overriding and overloading
- Examples of overriding and overloading
- Implementing overriding and overloading

Methods

In OOP, methods are the operations associated with a class:

- Methods are defined inside a class
- Methods give an object of a class its behaviour
- Sending a message to an object means calling or invoking a method in the object
- Methods are like functions, procedures or subroutines in other programming paradigms

The parts of a method

- Method name
- Return type
- Data type of parameters
- Order of Parameters
- Arity
- Signature
- Body

There are really just two main parts to a method:

- method header

- method body

The parts of a method

Method name

- This is the identifier used to invoke the method, should follow the identifier naming conventions of the implementing language

Return type

- The data type of the value returned by the method
  e.g. int M( ) { }, float P( ) { },
  void Q( ) { }, etc.
- Only one value can be returned at a time by each method

The parts of a method

Data Type of Parameters

- Each formal argument passed into the method has its own data type.
- A method may also not be passed any arguments and will therefore have an empty parameter list.

```
void method(int arg1, float arg2){}
```

- In this example, the method has two parameters arg1 and arg2 of data type int and float respectively.

The parts of a method

Order of Parameters

- The names given to the arguments passed to the method do not matter, only the order and the data type.
- Therefore void method(int arg1, float arg2){}and void method(int x, float y){} are equivalent, however void method(float a, int b){} or void method(double a, double b){} are not.

The parts of a method

Arity

- The arity of a method is the number of arguments or parameters that the method accepts.
- The arity is always given as a positive integer, as shown in the table.

Method	Arity
void fg(){ }	0
int o(int x, int y, int z){ }	3
public float loc(float pr){ }	1
void print(Student obj){ }	1
double k(double n, double m){ }	2
void s(int a, int b, int c, int d){ }	4
public Student(){ gpa = 0.0;}	0

The parts of a method

Signature

- The signature of a method is the name of the method and its parameter list including the data type, arity and order of the arguments but not the return type.

Body

- The body of a method contains the executable statements to perform the operation it is supposed to carry out.

The parts of a method

- Method name (setPrice)
- Return type (void)
- Data type of parameters (double)
- Parameter (d)
- Signature setPrice(double)
- Body {...}

C++

```
class Item
{
        private:
                double price;

        public:

        //method to set price a

        void setPrice(double d)
        {
                if (price > 0)
                        price = d;
                else
                        price = 0;
        }
};
```

The parts of a method

- Method name (setPrice)
- Return type (void)
- Data type of parameters (double)
- Parameter (d)
- Signature setPrice(double)
- Body {...}

Java

```
public class Item
{
private double price;

        //method to set price a

public  void setPrice(double d)
        {
                if (price > 0)
                        price = d;
                else
                        price = 0;
        }
};
```

Overriding

- Overriding occurs when more than one method share the same name, same signature and same return type, in an inheritance hierarchy
- If the parent class has a method called **M( )**, and the child class also has a method called **M( )**, then the child class method will override the parent class method
- However, there are mechanisms that allow the parent class' version of the overridden method to be called

Overloading

- Overloading occurs in a class when there is more than one method with the same name but each has a different signature
- If a class has several versions of a method **M( )**, for example **void M( )**, **void M(int x)**, **void M(float y, float z)**, then the method **M( )** is said to be overloaded
- Can also occur in an inheritance hierarchy

Differentiating between overriding and overloading

Characteristic	Overriding	Overloading
Method Name	Must be same	Must be same
Signature	Must be same	Must be different
Return Type	Must be same	Ignored
Data type of parameters	Must be same	Can be different
Order of parameters	Must be same	Can be different
Arity	Must be same	Can be different
Body	Can be different	Can be different

Examples of overriding and overloading

Example 1 - C++

Does these methods represent overriding, overloading or none of them?

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count;
                ++count;
        }

        return count;
}
```

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count << ",";
                ++count;
        }

        return count;
}
```

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count;
                cout <<
endl;
                ++count;
        }

        return count;
}
```

Examples of overriding and overloading

Example 1 - Java

Does these methods represent overriding, overloading or none of them?

```
public int ten()
{
        int count = 0;

        while(count < 10)
        {

System.out.print(count);
          ++count;
        }

        return count;
}
```

```
public int ten()
{
        int count = 0;

        while(count < 10)
        {

System.out.print(count);

System.out.print(“,”);
          ++count;
        }

        return count;
}
```

```
public int ten()
{
   int count = 0;

   while(count < 10)
   {
     System.out.println(count);
     ++count;
   }

   return count;
}
```

Examples of overriding and overloading

Example 1 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = overriding

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count;
                ++count;
        }

        return count;
}
```

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count << ",";
                ++count;
        }

        return count;
}
```

```
int ten()
{
        int count = 0;

        while(count < 10)
        {
                cout <<
count;
                cout <<
endl;
                ++count;
        }

        return count;
}
```

Examples of overriding and overloading

Example 2 - C++

Does these methods represent overriding, overloading or none of them?

```
int myst(int a, float b, string c)
{
        cout << "b is " << b;
        cout << "c is " << c;
        return a;
}
```

```
int myst(int c, float b, string a)
{
        cout << "a and b are
";
        cout << a << " " << b;
        return c;
}
```

```
int myst(int x, float y, string z)
{
        cout << "The 3 values
"
        << "are as follows: "
        << x << " " << y << "
"
        << z << endl;
        return x;
}
```

Examples of overriding and overloading

Example 2 - Java

Does these methods represent overriding, overloading or none of them?

```
public
int myst(int a, float b, string c)
{
   System.out.print(“b is ”+b);
   System.out.print(“c is ”+c);
   return a;
}
```

```
public
int myst(int c, float b, string a)
{
  System.out.print(
     “a and b are  ”);
  System.out.print( a + “ “ + b);
  return c;
}
```

```
public
int myst(int x, float y, string z)
{
                System.out.println(
                “The 3 values ”
                + “are as follows: “
                +  x + “ “ + y + “ “ +
z);
                return x;
}
```

Examples of overriding and overloading

Example 2 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = overriding

```
int myst(int a, float b, string c)
{
        cout << "b is " << b;
        cout << "c is " << c;
        return a;
}
```

```
int myst(int c, float b, string a)
{
        cout << "a and b are
";
        cout << a << " " << b;
        return c;
}
```

```
int myst(int x, float y, string z)
{
        cout << "The 3 values
"
        << "are as follows: "
        << x << " " << y << "
"
        << z << endl;
        return x;
}
```

Examples of overriding and overloading

Example 3 - C++

Does these methods represent overriding, overloading or none of them?

```
double deep(int p1)
{
        int ui;

        cout << “enter an
int:”;

        cin >> ui;

        return ui * p1;
}
```

```
void deeper(int p1, int p2)
{
        int ui;

        cout << “enter an
int:”;

        cin >> ui;

        cout << ui * p1 * p2;
}
```

```
string deepest
(int p1, int p2, int p3)
{
        int ui, res;

        cout << “enter an
int:”;

        cin >> ui;

        res = ui * p1 * p2 *
p3;

        return “Deepest”;
}
```

Examples of overriding and overloading

Example 3 - Java

Does these methods represent overriding, overloading or none of them?

```
public double deep(int p1)
{
  int ui;

  System.out.print(
   “enter an int:”);
  BufferedReader is =
   new BufferedReader(
    new InputStreamReader(
     System.in));
   try {
            String line;

            System.out.print(
       "Enter an int:");
            line = is.readLine();
            ui  =
Integer.parseInt(line);
            } catch (IOException e) {
   }
  return ui * p1;
}
```

```
public
void deeper(int p1, int p2) {
  int ui;
  System.out.print(
   “enter an int:”);
  BufferedReader is =
   new BufferedReader(
    new InputStreamReader(
     System.in));
   try {
            String line;

            System.out.print(
       "Enter an int:");
            line = is.readLine();
            ui  =
Integer.parseInt(line);
            } catch (IOException e) {
   }
  System.out.print(
   ui * p1 * p2);
}
```

```
public String deepest
(int p1, int p2, int p3)
{
  int ui, res;
  System.out.print(
   “enter an int:”);
  BufferedReader is =
   new BufferedReader(
    new InputStreamReader(
     System.in));
   try {
            String line;

            System.out.print(
       "Enter an int:");
            line = is.readLine();
            ui  =
Integer.parseInt(line);
            } catch (IOException e) {
   }
  res = ui * p1 * p2 * p3;
  return “Deepest”;  }
```

Examples of overriding and overloading

Example 3 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = none of them

```
public double deep(int p1)
{
        int ui;

        cout << "enter an
int:";

        cin >> ui;

        return ui * p1;
}
```

```
void deeper(int p1, int p2)
{
        int ui;

        cout << "enter an
int:";

        cin >> ui;

        cout << ui * p1 * p2;
}
```

```
string deepest
(int p1, int p2, int p3)
{
        int ui, res;

        cout << "enter an
int:";

        cin >> ui;

        res = ui * p1 * p2 *
p3;

        return "Deepest";
}
```

Examples of overriding and overloading

Example 4 - C++

Does these methods represent overriding, overloading or none of them?

```
char Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 'a';
        else
                return 'b';
}
```

```
int Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 1;
        else
                return 0;
}
```

```
double Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 5.2;
        else
                return 1.9;
}
```

Examples of overriding and overloading

Example 4 – (same answers apply to C++ and Java)

Do these methods represent overriding, overloading or none of them?

Answer = none

```
char Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 'a';
        else
                return 'b';
}
```

```
int Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 1;
        else
                return 0;
}
```

```
double Ossified(int level)
{
        int thickness=0;

        for(int i=0; i<level;
i++)
                thickness +=
i;

        if(thickness>12)
                return 5.2;
        else
                return 1.9;
}
```

Examples of overriding and overloading

Example 5 - C++

Does these methods represent overriding, overloading or none of them?

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;
}
```

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;

        cout << "auth. level:" <<
                authlevel <<
endl;
}
```

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;

        cout << "auth. level:" <<
                authlevel <<
endl;

        cout << "supv code:" <<
                supvcode <<
endl;
}
```

Examples of overriding and overloading

Example 5 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = overriding

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;
}
```

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;

        cout << "auth. level:" <<
                authlevel <<
endl;
}
```

```
void Display()
{
        cout << "emp no:" <<
                empno <<
endl;

        cout << "auth. level:" <<
                authlevel <<
endl;

        cout << "supv code:" <<
                supvcode <<
endl;
}
```

Examples of overriding and overloading

Example 6 - C++

Does these methods represent overriding, overloading or none of them?

```
void K(float J, int K, double L)
{




}
```

```
void K(int  J, float K, double L)
{




}
```

```
void K(int  J, double K, float L)
{




}
```

Examples of overriding and overloading

Example 6 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = overloading

```
void K(float J, int K, double L)
{




}
```

```
void K(int  J, float K, double L)
{




}
```

```
void K(int  J, double K, float L)
{




}
```

Examples of overriding and overloading

Example 7 - C++

Does these methods represent overriding, overloading or none of them?

```
void P(float J, int K, double L)
{




}
```

```
void P(float K, int J, double L)
{




}
```

```
void P(float J, int  L, double K)
{




}
```

Examples of overriding and overloading

Example 7 – (same answers apply to C++ and Java)

Does these methods represent overriding, overloading or none of them?

Answer = overriding

```
void P(float J, int K, double L)
{




}
```

```
void P(float K, int J, double L)
{




}
```

```
void P(float J, int  L, double K)
{




}
```

Implementing overriding and overloading

C++

Java

Overriding example – parent class

```
//Student class
#ifndef StudentH
#define StudentH

#include <iostream>
#include <string>
using namespace std;

class Student
{
        protected:
                string idNo;

        public:
                Student()
                {
                        idNo = "0000000";
                }
                virtual void Display()
                {
                        cout << "Id:" << idNo
<< endl;
                }

};
#endif
```

C++ requires the keyword virtual to be in the parent class prefacing the overridden method

```
//Student class
public class Student
{
        protected String idNo;

        public Student()
        {
                idNo = "0000000";
        }

        public void Display()
        {
                System.out.println("Id:"  + idNo );

        }
}
```

Display( ) is a method in the parent class

Implementing overriding and overloading

C++

Overriding example – child class

Java

```
//ArchitectureStudent class
#ifndef ArchitectureStudentH
#define ArchitectureStudentH

#include "Student.h"
#include <iostream>
#include <string>
using namespace std;

class ArchitectureStudent : public Student
{
        private:
                double plotterCreditBalance;
        public:
                ArchitectureStudent(string id, double pc)
                {
                        idNo = id;
                        plotterCreditBalance = pc;
                }
                void Display()
                {
                        cout << "Id:" << idNo << endl;
                        cout << "Plotter credit
balance:"
                         << plotterCreditBalance <<
endl;
                }
};
#endif
```

```
//ArchitectureStudent class
class ArchitectureStudent extends Student
{
        private double plotterCreditBalance;

        public ArchitectureStudent(String id, double pc)
        {
                idNo = id;
                plotterCreditBalance = pc;
        }
        public void Display()
        {
                System.out.println("Id:" + idNo);
                System.out.println("Plotter credit
balance:"
                + plotterCreditBalance);
        }
}
```

Now the Display( ) method is overridden in the child class

Implementing overriding and overloading

C++

Overriding example – driver with main( )

Java

```
#include "Student.h"
#include "ArchitectureStudent.h"
using namespace std;

int main ()
{

        Student John;
        John.Display();

        ArchitectureStudent Sue("1234567", 150.0);
        Sue.Display();

        return 0;
}
```

Calling the Display( ) method in the parent class

```
public class Driver
{

    public static void main(String[] args)
    {
            Student John = new Student();
            John.Display();

            ArchitectureStudent Sue = new
ArchitectureStudent("1234567",  150.0);
            Sue.Display();
        }

}
```

Calling the Display( ) method in the child class

Implementing overriding and overloading

C++

Overriding example – calling the parent class' method

Java

```
//ArchitectureStudent class
#ifndef ArchitectureStudentH
#define ArchitectureStudentH

#include "Student.h"
#include <iostream>
#include <string>
using namespace std;

class ArchitectureStudent : public Student
{
        private:
                double PlotterCreditBalance;
        public:
                ArchitectureStudent(string id, double pc)
                {
                        idNo = id;
                        PlotterCreditBalance = pc;
                }
                void Display()
                {
                        Student::Display();
                        cout << "Plotter credit
balance:"
                         << PlotterCreditBalance <<
endl;
                }
};
#endif
```

```
//ArchitectureStudent class
class ArchitectureStudent extends Student
{
        private double PlotterCreditBalance;
        public ArchitectureStudent(String id, double pc)
        {
                idNo = id;
                PlotterCreditBalance = pc;
        }
        public void Display()
        {
                super.Display();
                System.out.println("Plotter credit
balance:"
                 + PlotterCreditBalance);
        }
}
```

Invoking the parent class version of the Display( ) method from a child class

C++ - class name plus binary scope operator plus method

Java - keyword *super* plus . plus method

Implementing overriding and overloading

C++

Overriding example – sample output from program

Java

```
Id:0000000
Id:1234567
Plotter credit balance:150
```

```
Id:0000000
Id:1234567
Plotter credit balance:150.0
```

When Display( ) is called from the parent class object it prints the Id number of the student.

When the overridden Display( ) is called from the child class object, it prints both the Id number and the student's plotter credit balance

Overriding will come in very handy in subsequent lectures when we look at polymorphism

Implementing overriding and overloading

C++

overloading example

Java

```
#include <string>
using namespace std;

//Calculate class
class Calculate
{
  public:
        //add two ints, returns an int
        int sum(int x, int y)
        {
                return x + y;
        }

        //add two doubles, returns a double
        double sum(double x, double y)
        {
                return x + y;
        }

        //add two strings, returns a string
        string sum(string x, string y)
        {
                return x + y;
        }
};
```

```
//Calculate class
public class Calculate
{
        //add two integers, returns an integer
        public int sum(int x, int y)
        {
                return x + y;
        }

        //add two doubles, returns a double
        public double sum(double x, double y)
        {
                return x + y;
        }

        //add two strings, returns a string
        public String sum(String x, String y)
        {
                return x + y;
        }
}
```

In this example the Sum( ) method is overloaded

Implementing overriding and overloading

C++

overloading example

Java

```
#include "Calculate.h"
#include <iostream>
using namespace std;

int main (int argc, char *argv[])
{

        Calculate T;

        cout << "5 + 6 is ";
        cout << T.sum(5,6);
        cout << endl;

        cout << "9.8 + 2.4 is ";
        cout << T.sum(9.8,2.4);
        cout << endl;

        cout << "Hi + There is ";
        cout << T.sum("Hi","There");
        cout << endl;

        return 0;
}
```

```
public class Driver
{
        public static void main(String[] args)
        {
                Calculate T = new Calculate();

                System.out.print("5 + 6 is ");
                System.out.println(T.sum(5,6));

                System.out.print("9.8 + 2.4 is ");

        System.out.println(T.sum(9.8,2.4));

                System.out.print("Hi + There is ");

        System.out.println(T.sum("Hi","There"));
        }
}
```

Note that in this example, the Sum( ) method is called three times. Each time it is called, the compiler matches the most appropriate version of the method with the call at compile time. If no appropriate match is found then a compile time error is generated.

Implementing overriding and overloading

C++

overloading example – sample output from program

Java

```
5 + 6 is 11
9.8 + 2.4 is 12.2
Hi + There is HiThere
```

```
5 + 6 is 11
9.8 + 2.4 is 12.2
Hi + There is HiThere
```

In the first call, the compiler invokes the integer version of Sum( ) to compute Sum(5,6).

In the second call, the compiler invokes the double version of Sum( ) to compute Sum(9.8,2.4).

In the third call, the compiler invokes the string version of Sum( ) to compute Sum("Hi","There").

The compiler takes the signature and return type into account when trying to determine which, if any, of the overloaded methods to call