Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

CET1042B: Object Oriented Programming with C++ and Java

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

S. Y. B. TECH. COMPUTER SCIENCE AND ENGINEERING
(CYBERSECURITY AND FORENSICS)

CET1042B: Object Oriented Programming with C++ and Java

Teaching Scheme
Theory: 2 Hrs. / Week

Credits: 02 + 02 = 04
Practical: 4 Hrs./Week

Course Objectives

1) **Knowledge:** (i) Learn object oriented paradigm and its fundamentals.
2) **Skills:** (i) Understand Inheritance, Polymorphism and dynamic binding using OOP.
   (ii) Study the concepts of Exception Handling and file handling using C++ and Java.
3) **Attitude:** (i) Learn to apply advanced concepts to solve real world problems.

Course Outcomes

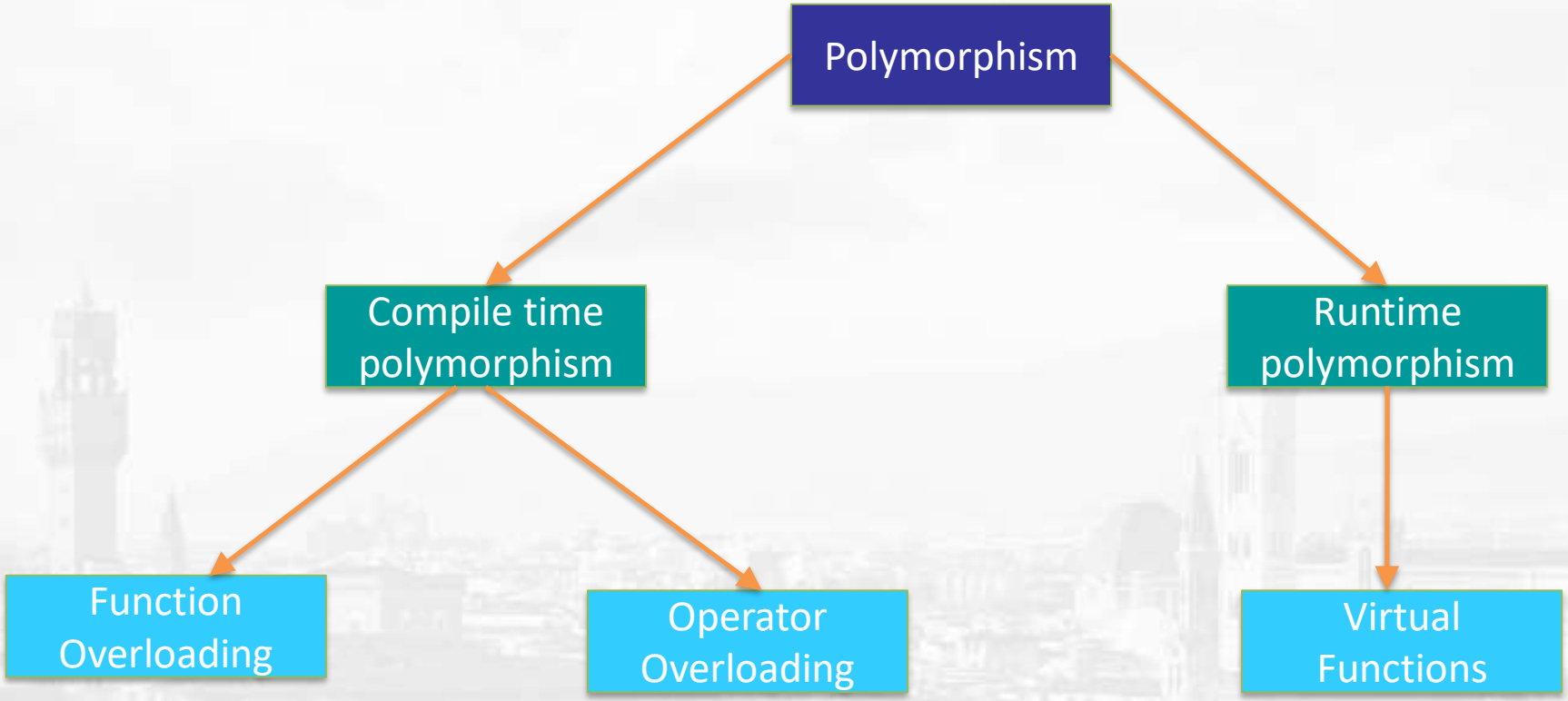
1) Apply the basic concepts of Object Oriented Programming to design an application.
2) Make use of Inheritance and Polymorphism to develop real world applications.
3) Apply the concepts of exceptions and file handling to store and retrieve the data.
4) Develop efficient application solutions using advanced concepts.

Assignment 3

Implementation of Polymorphism using C++ and JAVA.

Getting Introduced with Polymorphism

Polymorphism



Virtual Functions

- Virtual function is a member function in the base class that you redefine in a derived class.
- To make a function virtual, we write the keyword **virtual** before the function definition.
- It is used to tell the compiler to perform dynamic linkage or late binding on the function.
- There is a necessity to use the single pointer to refer to all the objects of the different classes.
- So, we create the pointer to the base class that refers to all the derived objects.
- A virtual member function in a base class automatically becomes virtual in all of its derived classes.
- When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.

Virtual Function Example

```
class Animals {
    public:
      virtual void sound()
{
            cout << "This is parent class" << endl;
    }
};

class Dogs : public Animals
{
    private:
    virtual void sound() {
            cout << "Dogs bark" << endl;
    }
};
```

```
main() {
Animals  *a;
Dogs  d;
a = &d;
a -> sound();  // late binding
}
```

Output

Dogs bark

We can also call private function of derived class from a base class pointer by declaring that function in the base class as virtual.

Pure Virtual Function

- **It** is a virtual function which has no definition related to base class.
- Pure virtual function is a virtual function which is not used for performing any task.
- It only serves as a placeholder.
- The "**do-nothing**" function is known as a **pure virtual function**.
- A pure virtual function is a function declared in the base class that has no definition relative to the base class. Also called as **abstract functions**
- To create a pure virtual function, we assign a value **0** to the function
- **e.g. virtual void sound() = 0;**
- Tells compiler that there *is no* implementation
- A class containing the pure virtual function cannot be used to declare the objects of its own, such classes are known as **abstract base** classes.

Pure Virtual Function Example

```
#include <iostream>
using namespace std;
class Base
{
   public:
   virtual void show() = 0;
};
class Derived : public Base
   {
   public:
   void show()
 {
     cout << "Derived class is derived from the base class." ;
      }
};
```

```
int main()
{
   Base *bptr;
   //Base b;
   Derived d;
   bptr = &d;
   bptr->show();
   return 0;
}
```

Output:

Derived class is derived from the base class.

Note: Here, the base class contains the pure virtual function.
Therefore, the base class is an abstract base class. We cannot create the object of the base class.

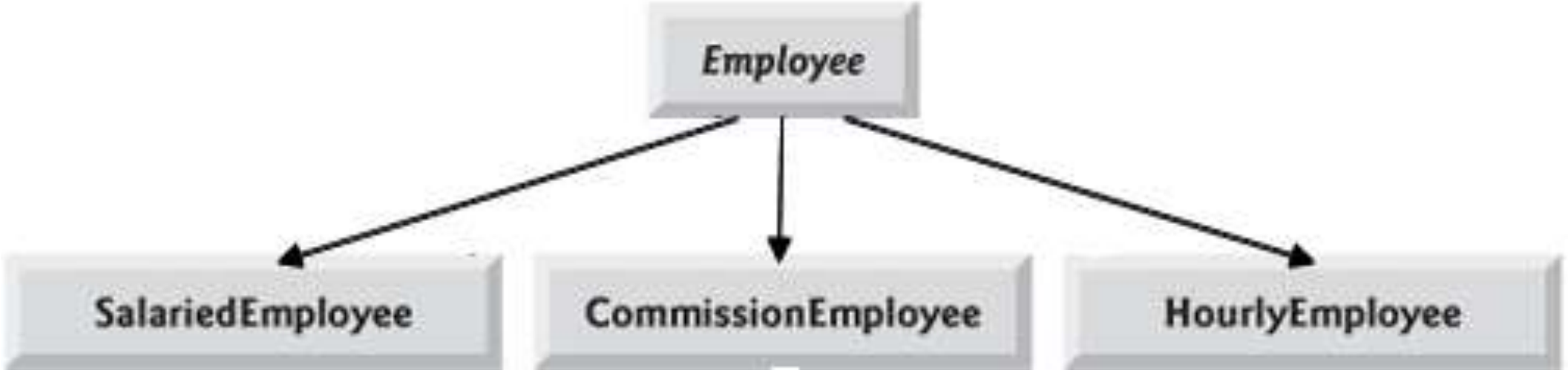
FAQs

- Difference Between Virtual and Pure Virtual Function.
- What are virtual functions in C++?
- How do you define a pure virtual function?
- What are the characteristics of virtual function?

Problem Statement_Assignment 3A_using C++

- Write a C++ program with base class Employee and three derived classes namely
  i. salaried employees
  ii. commission_employees and
  iii. hourly employees
- Declare **calculate_salary()** as a pure virtual function in base class and define it in respective derived classes to calculate salary of an employee.
- The company wants to implement an Object Oriented Application that performs its payroll calculations polymorphically.

Employee hierarchy class diagram



Algorithm: Payroll System Using Polymorphism

1. Start
2. Create a super class Employee
3. Create Subclass Salaried_Employee
4. Create Subclass Commission_Employee
5. Create Subclass Hourly_Employee
6. Define data members and member functions associated with respective classes.
7. Implement function overloading on calculate_salary() member function
8. Calculate and display earnings of each employee based on their type.
9. Stop

Table 1. Polymorphic representation for the Employee hierarchy classes

Class / Function Name	earnings	Data Members
Employee	abstract	firstName, lastName Social Security Number: SSN
Salaried-Employee	weeklySalary	Salaried Employee: firstName, lastName Social Security Number: SSN Weekly Salary: weeklySalary

Hourly-Employee	if (hours<=40)
	wage*hours
	else if (hours>40)
	{
	40*wage+(hours-40)*wage*1.5
	}
		Hourly Employee: firstName, lastName
		Social Security Number: SSN
		Hourly wage: wage;
		hours worked: hours

Commission-Employee	commissionRate*grossSales	Commission Employee: firstName, lastName Social Security Number: SSN Gross sales: grossSales; Commission rate: commissionRate

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Method Overloading and Method Overriding in Java

Method Overloading

- Multiple methods with the same name to perform different operations on different parameters.
  - Overloading is having multiple methods in the same class with the same name, but accept different types of parameters.
  - For instance:

```
public double add(double num1, double num2) {
        return num1 + num2;
}

public String add(String str1, String str2) {
        return str1 + str2;
}
```

  - Even though both of these methods are named `add`, they perform different operations on different parameters.

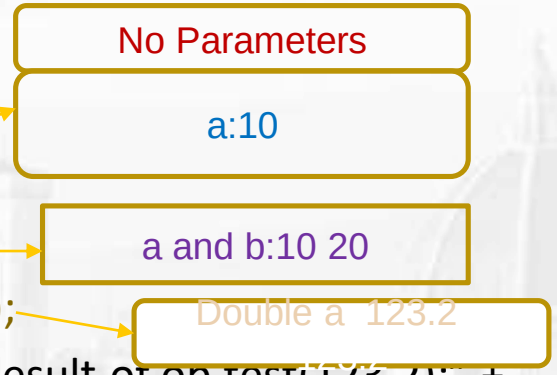
Method Overloading

- When we call a method, the compiler must determine which of the methods to use through a process called binding.
  - Java binds methods by matching a method's signature to how it is called.
    - A method's signature consists of its name and the data types of its parameters.
    - The signatures of the two previous methods are:
      - add(double, double)
      - add(String, String)
    - So the java compiler can tell which method to be used based on how it was called.
    - Note, that you cannot have methods with the same name and same data types for parameters EVEN IF THEY HAVE A DIFFERENT return type.

Simple Example of Method Overloading

```
class OverloadDemo
{
void test() {
  System.out.println("No parameters");
}
void test(int a) {
System.out.println("a:"+a);
}
void test(int a, int b){
System.out.println("a and b:"+a+ " "+b);
}
double test(double a){
System.out.println("double a: " +a);
return a*a;
}
}
```

```
class Overload {
Public static void main(String args[])
{
 OverloadDemo ob= newOverloadDemo();
 double result;
ob.test();
ob.test(10);
ob.test(10,20);
result= ob.test(123.2);
System.out.println("Result of ob.test(123.2): " +
result);
}
}
```



Overloading Constructors

- Constructors can be overloaded like methods.
- The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

Example of Constructor Overloading

```
Class Box{
  double width, height,depth;
Box(double w, double h, double d){
     width=w;
      height=h;
      depth = d;
}
Box(){
    width=-1;
     height = -1;
     depth= -1;
}
Box(double len) {
   width=height=depth=len;
}
double volume() {
   return width * height * depth;
}
}
```

```
Class OverloadCons{
   public static void main(String args[]){
     Box mybox1= new Box(10,20,15);
     Box mybox2 = new Box();
     Box mycube= new Box(7);
     double vol;
     // get volume of first box
     vol= mybox1.volume();
     System.out.println("Volume of mybox1 is " +vol);
     //get volume of second box
      vol= mybox2.volume();
      System.out.println("Volume of mybox2 is" +vol);
// get volume of cube
   vol=mycube.volume()
   System.out.println("Volume of mycube is" +vol);
}
}
```

3000.0

-1.0

343.0

Using Objects as Parameters

```
class Test{
    int a,b;
    Test(int i, int j)
    {
        a=i;
        b=j;
 }
 // return true if o is equal to the
invoking object
boolean equal(Test o)   {
   if(o.a = = this.a && o.b==this.b)
return true;
else  return false;
}
}
```

```
Class PassOb {
Public static void main (string args[]){
   Test ob1= new Test(100, 22);
    Test ob2= new Tess(100,22);
    Test ob3 = new Test(-1, -1);

System.out.println("ob1 == ob2: "
+ob1.equal(ob2));
```

Output-> ob1 == ob2 : true

```
System.out.println("ob1 == ob3:" +
ob1.equal(ob3));
}
```

Output-> ob1 == ob3: false

```
}
```

Problem Statement_Assignment 3B_Implementing Method Overloading_using Java

Define a class Shapes as

1. Circle
2. square and
3. rectangle.

Find the area of these shapes using constructor overloading and method overloading.

Method Overriding

- If subclass (child class)has the same method as declared in the parent class, it is known as method overriding.

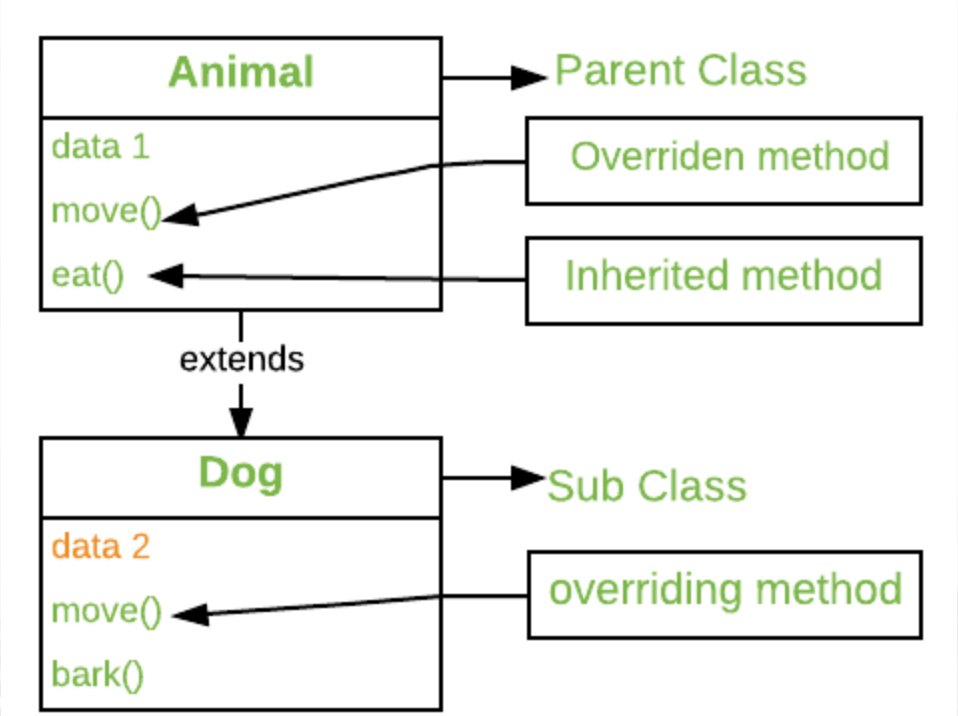
- Usage

1. It is used to provide specific implementation of method that is already provided by its super class.
2. Method overriding is used for runtime polymorphism.

- Rule

1. Method must have same name as in the parent class.
2. Method must have same parameter as in the parent class.

Method Overriding



The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed.

Example of Method Overriding

```
Class Figure {
    double dim1;
    double dim2;
Figure(double a, double b)   {
    dim1 = a;
    dim2 = b;
}
double area()   {
 System.out.println("Area for Figure is
undefined.");
 return 0;
}
}
Class Rectangle extends Figure {
   Rectangle(double a, double b) {
      super(a,b);
}
//override area for rectangle
 double area()
{
 System.out.println("Inside Area for Rectangle.");
 return dim1 * dim2;
}
}
```

```
Class Triangle extends Figure {
 Triangle(double a, double b) {
       super(a, b);
}
double area() {
 System.out.println("Inside area for Triangle.");
 return dim1 * dim2 /2;
 }
}
class FindAreas {
 public static void main(Sting args[]) {
    Figure f= new Figure(10,10);
    rectangle r= new Rectangle(9,5);
    Triangle t = new Triangle(10, 8);
     Figure figref;
     figref = r;
     System.out.println("Area is" +figref.area());
    figref = t;
    System.out.println("Area is" + figref.area());
    figref = f;
    System.out.println("Area is"+ figref.area());
}
}
```

45

40

0

Problem Statement_Assignment 3C_Implementing Method Overriding_using Java

- Create a parent class Hillstations with the methods location() and famousfor().
- Create three subclasses by hill station names e.g. Manali, Shimla etc.
- Subclasses extend the superclass and override methods location() and famousfor().
- Call the methods location() and famousfor() by the Parent class, i.e. Hillstations class.
- It should refer to the base class object and the base class method overrides the superclass method, base class method is invoked at runtime.

Key learnings

- Constructor overloading is somewhat similar to method overloading
- Constructor overloading is used for different ways of initializing an object
- Method overloading is used when there are different definitions of a method based on different parameters.
- For method overriding, both the superclass and the subclass must have the same method name, the same return type and the same parameter list.
- We cannot override the method declared as final and static .
- We should always override abstract methods of the superclass

Practice Assignments

1. A drawing program needs to display many shapes, including new shape types that the programmer will add to the system after writing the drawing program. The drawing program might need to display shapes, such as Circles, Triangles, Rectangles or others, that derive from abstract superclass Shape. The drawing program uses Shape variables to manage the objects that are displayed. To draw any object in this inheritance hierarchy, the drawing program uses a superclass Shape variable containing a reference to the subclass object to invoke the object's draw method. This method is declared abstract in superclass Shape, so each concrete subclass must implement method draw in a manner specific to that shape. Each object in the Shape inheritance hierarchy knows how to draw itself. The drawing program does not have to worry about the type of each object or whether the drawing program has ever encountered objects of that type.

Practice Assignments

2. A class Rectangle is derived from class Quadrilateral, as a Rectangle object is a more specific version of a Quadrilateral object. The operation (e.g., calculating the perimeter or the area) that can be performed on a Quadrilateral object and can also be performed on a Rectangle object. These operations can also be performed on other Quadrilaterals, such as Squares, Parallelograms and Trapezoids. Implement the polymorphism which occurs when a program invokes a method through a superclass variable—at execution time, the correct subclass version of the method is called, based on the type of the reference stored in the superclass variable.

Practice Assignments

3. Write a C++ code to implement the concept of inheritance for Vehicles. You are required to implement inheritance between classes. You have to write four classes i.e. one superclass, two sub classes and one driver class.

   Vehicle is the super class whereas Bus and Truck are sub classes of Vehicle class. Transport is a driver class which contains main method.

4. Write program will demonstrate example of hierarchical inheritance to get square and cube of a number in C++ programming language.

5. Write a program to have information about books in a library. Create another class with student information. Derive a class for issue of books from the classes ‘book’ and ‘student’. Derive another class for return of books from the class ‘issue’. Calculate fine for all the students who have over dues.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Thank You!

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

References

- https://www.tutorialspoint.com/java/java_inheritance.htm
- https://beginnersbook.com/2013/03/inheritance-in-java/
- https://www.geeksforgeeks.org/java-and-multiple-inheritance/
- https://www.mygreatlearning.com/blog/inheritance-in-java/
- https://www.w3schools.com/java/java_inner_classes.asp

MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Thank You!!