Object Technology
PS02CDCA33, PGDCA II Semester, 2021

Dr. Priti Srinivas Sajja

Professor, Department of Computer Science,
Sardar Patel University,
Vallabh Vidyanagar, Gujarat, India.

Introduction

Name:	**Dr. Priti Srinivas Sajja**
Email :	priti@pritisajja.info
Mobile :	+91 9824926020
URL:	**http://pritisajja.info**
Academic Quali.:	Ph. D in Computer Science
Thesis Title:	Knowledge-Based Systems for Socio-Economic Rural Development(2000)
Area of Interest:	Artificial Intelligence
Publications:	216 in International/ National Conferences, Journals & Books

Unit 3: Classes, Objects and Methods

- Class, Object, Object reference,
- Constructor, Constructor Overloading, Method Overloading,
- New operator, this and static keyword,
- Passing and Returning object form Method,
- finalize() method, Access Control Modifiers, Nested class, Inner class

Employee Payslip

- **Class in java**

```
        public class Emp {
                int eno;
                String ename;
                double bpay;
}
```

- **Method in java**

```
   void print () {
   System.out.println(" ------------------------------------");
   …
  }
```

Employee Payslip

- **Constructor in java (method overloading) using this keyword**

```
Emp (int eno, String ename, double bpay)  // a constructor
{ this.eno= eno;  this.ename= ename;   this.bpay= bpay;   }

Emp ()  // default constructor
{ this.eno= 0;  this.ename= "  ";   this.bpay= 0.0;   }

Emp (int eno, String ename)   // one more constructor
{ this.eno= eno;      this.ename= ename;  this.bpay= 0.0;   }
```

Polymorphism: One name, three implementations

Employee Payslip

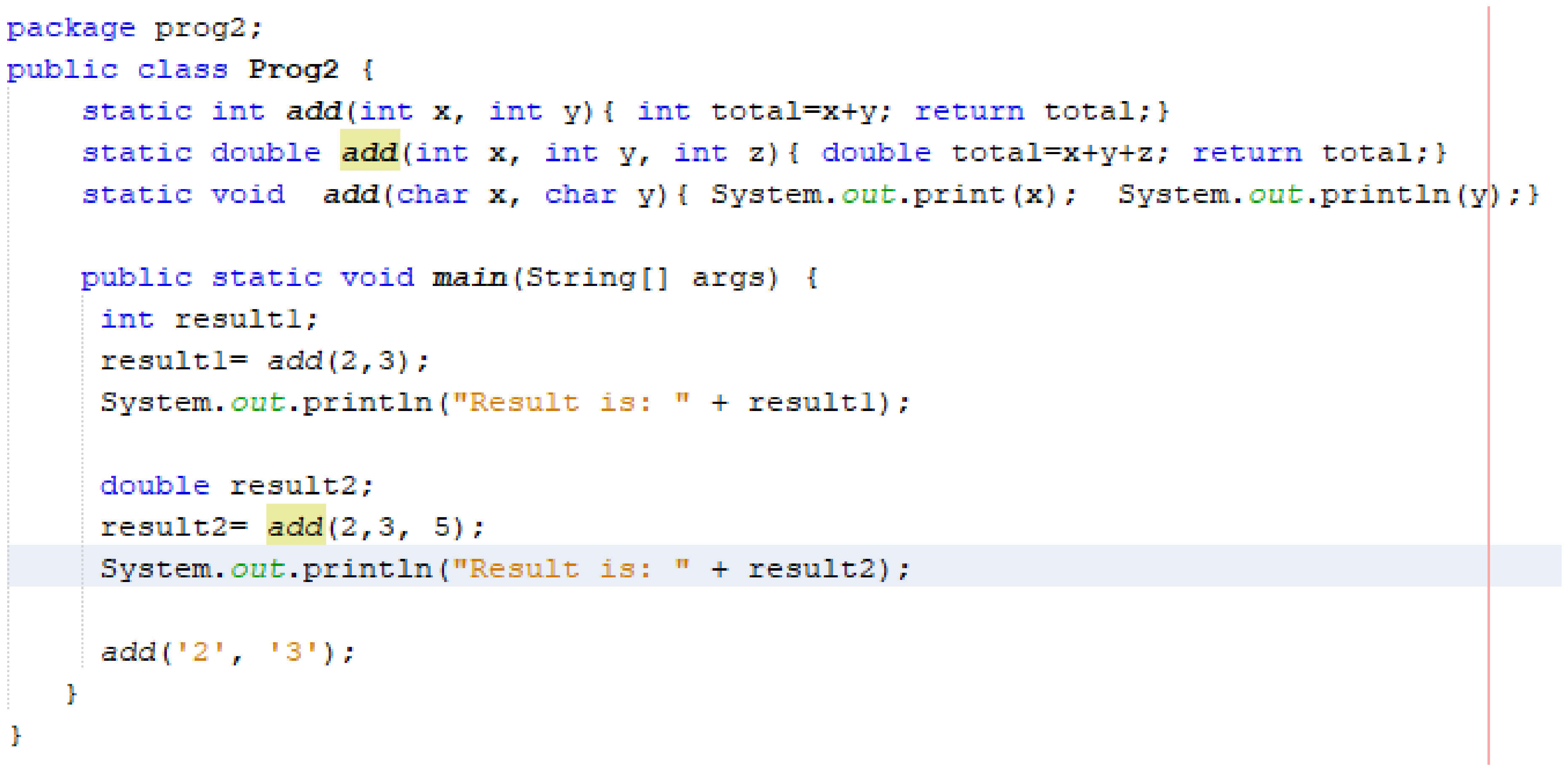
- **Defining objects in a class using New**

```
Emp e1 = new Emp(1, "XXX XXX ", 10000.00);
Emp e2 = new Emp(2, "YYYYYYY ", 50000.00);
```

- **Referring methods/attributes on objects**

```
e1.print();
e2.print();
e1. eno;
e2. eno;
```

Static Method and Passing and Returning Form Method



```
package prog2;
public class Prog2 {
    static int add(int x, int y){ int total=x+y; return total;}
    static double add(int x, int y, int z){ double total=x+y+z; return total;}
    static void  add(char x, char y){ System.out.print(x);  System.out.println(y);}

    public static void main(String[] args) {
     int result1;
     result1= add(2,3);
     System.out.println("Result is: " + result1);

     double result2;
     result2= add(2,3, 5);
     System.out.println("Result is: " + result2);

     add('2', '3');
   }
}
```

Final Variable in Java

- **final int SIZE=10;**
- **Initialization** is must.
- The value given becomes **constant**.
- It is a tradition to write final variables in **CAPITAL**.
- By default other methods and variables can be **overridden**, but not final.
- Final variable behave like **class variable** and **does not occupy space** on object.

Final Class in Java

- **final class A {….}**
- **Extension** is not possible. That is subclasses can not be extended.

```
final class A{
……….}

Class B extends A{
……….}
gives error.
```

Final Method in Java

- **class bird{**
- **final void fly()**
- { System.out.println("This will not change...");}
- }

- **class nonflybird extends bird{**
- void fly(){ System.out.prinltn("ERROR....");}
- }

Garbage Collection in Java

- Java does **garbage collection automatically**.
- **When no reference of an object exists, it is no longer needed, and memory occupied by it can be reclaimed.**
- **No explicit need** of manual garbage collection.
- The garbage collection done periodically by java run time system.

Finalize() in Java

- Java, while run time clears memory and does **automatic garbage collection**.
- But **non-object references** are still there in memory(eg. Window systems fonts), to clear this, the finalize() methods is used.
- That is when object automatically calls the garbage collector, **you may choose to do some actions**.
- **finalize() { };**
- Can be included in any method. Java calls this method when an object is going to be reclaimed/recycled.

Visibility Control: Public:

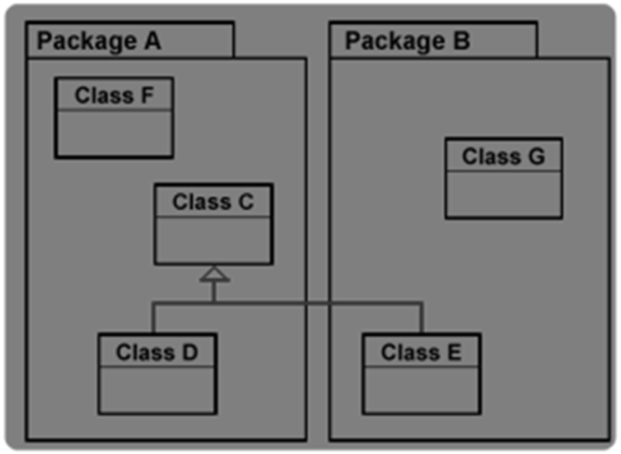
- Any variable or method defined as **public is visible to the entire class**.
- The main() method is called from outside the program→ Java runtime system.
- It is also visible to **all the classes outside the class** in all the packages
- When **no access specifier** is used then by default the member of class is public within its own package, but **not be accessed outside of its package**.

Visibility Control: Friendly:

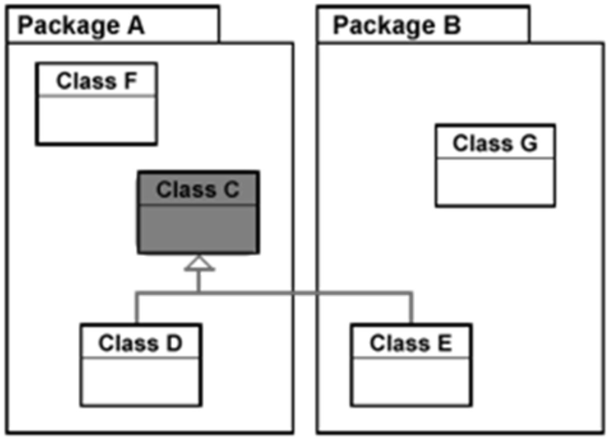
- It is by default
- It is visible to the classes in which it is defined

Visibility Control: Private:

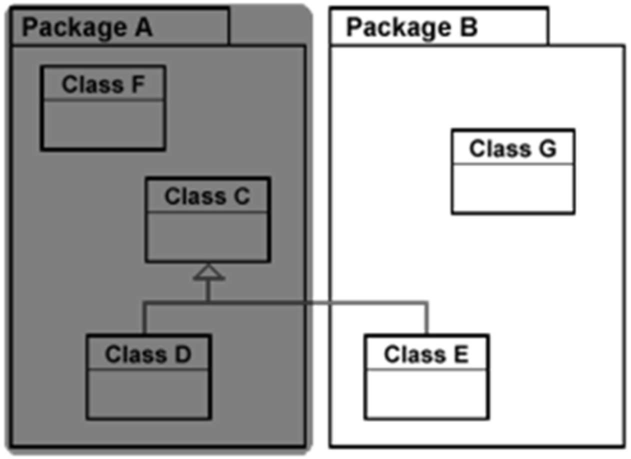
- It is visible to the **class and its all subclasses**
- We can not override a non-private method in a subclass and then make it private.
- Private and Protected together→ visible in all subclasses only(not in non-class) regardless of any package.



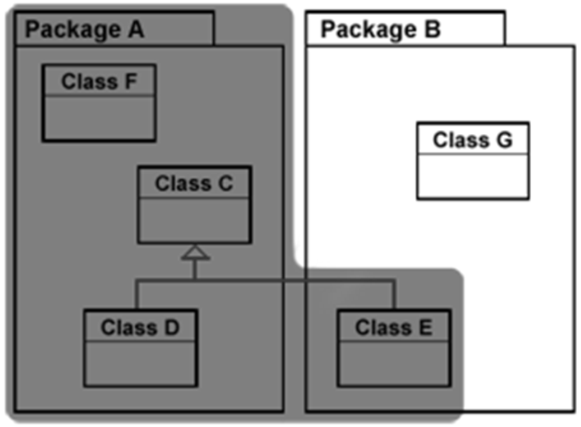
Public



Private



Default



Protected

Points to Remember …

- **Public grants access to anyone.**
- **Private denies access to everyone** except code within that same class.
- **Protected provides access to all code in the same packages and to subclasses** in different packages.
- The **default access restricts access to within the same package**.

Nested Class and Inner Class

- **Java inner class** or nested class is a class which is declared inside the class or interface.
- Additionally, it can access all the members of outer class including private data members and methods.

```
class Java_Outer_class{
 //code
 class Java_Inner_class{
  //code
 }
}
```

Advantages of Inner Class

- Nested classes represent a special type of relationship that is **it can access all the members (data members and methods) of outer class** including private.
- Nested classes are used **to develop more readable and maintainable code** because it logically group classes and interfaces in one place only.
- **Code Optimization**: It requires less code to write.

Acknowledgement

Patrick Naughton and Herbert Schildt, The Complete Reference Java 2, Seventh, Tata McGraw Hill Pub., 2007

Javapoint.com

W3schools.com