COJ :: Class Design & Encapsulation

TalentSprint

Licensed To Skill

Version 1.0.4

Encapsulation

The content in this presentation is aimed at learners to learn :

- Explain the need and importance of Encapsulation
- Applying encapsulation for java classes

Encapsulation

Create a class Rectangle having the following :
Rectangle:

Instance variables: length: double
breadth: double

Instance methods : getArea() : double
getPerimeter() :double

Provide proper constructors for all classes.
Create a general class “MyClass”. In this class create two objects of Rectangle class and then set the length and breadth attributes for each object ,compute their area and perimeter and display them

Encapsulation

Access Modifiers

Public	Keyword applied to a class, makes it available / visible everywhere.
	Keyword applied to a method or variable, makes it completely visible.
Private	Fields or Methods of a class are visible only within that class.
	Members are not visible within subclasses, and are not inherited.

Encapsulation

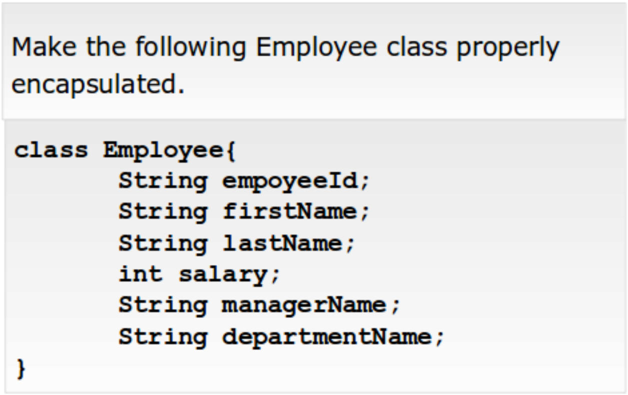
Visibility

```
class Rectangle {
    private double length;
    public breadth;
            // Constructor
    public Rectangle (double length, double breadth) {
        this. length = length;
this. breadth = breadth;
    }
            //Methods to return circumference and area
    public double getPerimeter() {
        return 2∗ (length + breadth);
    }
    public double getArea() {
        return length ∗ breadth;
    }
}
```

Encapsulation

```
class MainClass {
    public static void main(String args[]){
        Rectngle rectangle1 = new Rectnagle(10,10);
rectangle1. length = 10;   //  Error.  Private cannot be accessed
                                         from out side the class
rectangle1. breadth = 10;  // No Error.  public can be accessed
    from
                                         outside the class.
        System.out.pritnln(rectangle1.getPerimeter());
System.out.pritnln(rectangle1.getArea());
    }
}
```

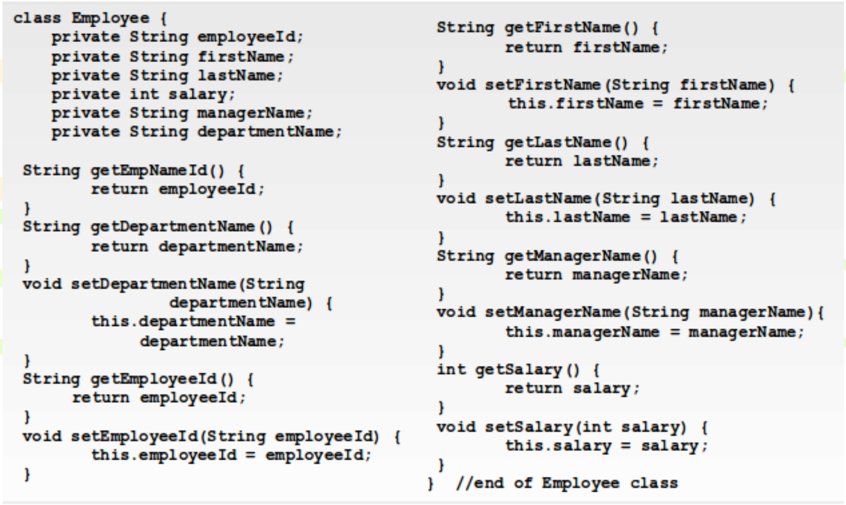
Encapsulation



Make the following Employee class properly encapsulated.

```
class Employee{
        String empoyeeId;
        String firstName;
        String lastName;
        int salary;
        String managerName;
        String departmentName;
}
```

Encapsulation



```
class Employee {
    private String employeeId;
    private String firstName;
    private String lastName;
    private int salary;
    private String managerName;
    private String departmentName;

 String getEmpNameId() {
        return employeeId;
 }
 String getDepartmentName () {
        return departmentName;
 }
 void setDepartmentName(String
                departmentName) {
        this.departmentName =
             departmentName;
 }
 String getEmployeeId() {
      return employeeId;
 }
 void setEmployeeId(String employeeId) {
        this.employeeId = employeeId;
 }
 String getFirstName() {
        return firstName;
 }
 void setFirstName(String firstName) {
        this.firstName = firstName;
 }
 String getLastName() {
        return lastName;
 }
 void setLastName(String lastName) {
        this.lastName = lastName;
 }
 String getManagerName() {
        return managerName;
 }
 void setManagerName(String managerName){
        this.managerName = managerName;
 }
 int getSalary() {
        return salary;
 }
 void setSalary(int salary) {
        this.salary = salary;
 }
}  //end of Employee class
```

Encapsulation

Encapsulation

- Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods.
- If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class.
- Encapsulation is also referred to as data hiding

Encapsulation

Benefits of Encapsulation

- Ability to modify our implemented code without breaking the code of others who use our code.
- Helps in protecting against accidental or wrong usage.
- Gives maintainability, flexibility and extensibility to our code.

Encapsulation