Sample code examples

**Example code 1**

**package** sectionC1;

**import** java.util.\*;

**public** **class** demo34 {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter elements in array");

**int** number;

number=sc.nextInt();

ArrayList<String> people=**new** ArrayList<>();

**for**(**int** i=0;i<number;i++)

{

people.add(sc.next());

}

people.remove(0);

people.remove(1);

**for**(String s:people)

{

System.***out***.println(s);

}

/\*

LinkedList<Integer> lt=new LinkedList<>();

lt.add(234);

lt.add(456);

lt.add(678);

for(int i :lt)

{

System.out.println(i);

}

\*/

}

}

**Example code 2**

// Define an interface named RemoteWork

interface RemoteWork {

void workRemotely();

}

// Define an abstract class named Employee

abstract class Employee {

String name;

String position;

Employee(String name, String position) {

this.name = name;

this.position = position;

}

// Abstract method (does not have a body)

abstract void performDuties();

// Regular method

void getDetails() {

System.out.println("Name: " + name);

System.out.println("Position: " + position);

}

}

// Concrete class SoftwareEngineer extends Employee and implements RemoteWork

class SoftwareEngineer extends Employee implements RemoteWork {

SoftwareEngineer(String name) {

super(name, "Software Engineer");

}

// Implementation of abstract method from Employee

@Override

void performDuties() {

System.out.println(name + " is writing code and fixing bugs.");

}

// Implementation of method from RemoteWork interface

@Override

public void workRemotely() {

System.out.println(name + " is working remotely.");

}

}

public class Main {

public static void main(String[] args) {

// Create an instance of SoftwareEngineer

SoftwareEngineer engineer = new SoftwareEngineer("Alice");

// Call methods from Employee and RemoteWork

engineer.getDetails();

engineer.performDuties();

engineer.workRemotely();

}

}

**Example code 3**

Composition: A Library class contains Book objects. When the Library is destroyed, the Book objects are also destroyed.

Aggregation: A Department class contains Professor objects. The Professor objects can exist independently of the Department.

// Class representing a Book

class Book {

private String title;

public Book(String title) {

this.title = title;

}

public String getTitle() {

return title;

}

}

// Class representing a Library with composition

class Library {

private String name;

private List<Book> books;

public Library(String name) {

this.name = name;

this.books = new ArrayList<>();

}

public void addBook(Book book) {

books.add(book);

}

public void showBooks() {

System.out.println("Library: " + name);

for (Book book : books) {

System.out.println("Book: " + book.getTitle());

}

}

}

// Class representing a Professor

class Professor {

private String name;

public Professor(String name) {

this.name = name;

}

public String getName() {

return name;

}

}

// Class representing a Department with aggregation

class Department {

private String name;

private List<Professor> professors;

public Department(String name) {

this.name = name;

this.professors = new ArrayList<>();

}

public void addProfessor(Professor professor) {

professors.add(professor);

}

public void showProfessors() {

System.out.println("Department: " + name);

for (Professor professor : professors) {

System.out.println("Professor: " + professor.getName());

}

}

}

// Main class to demonstrate composition and aggregation

public class Main {

public static void main(String[] args) {

// Demonstrate composition

Library library = new Library("City Library");

Book book1 = new Book("To Kill a Mockingbird");

Book book2 = new Book("1984");

library.addBook(book1);

library.addBook(book2);

library.showBooks();

// Demonstrate aggregation

Department department = new Department("Computer Science");

Professor professor1 = new Professor("Dr. Smith");

Professor professor2 = new Professor("Dr. Johnson");

department.addProfessor(professor1);

department.addProfessor(professor2);

department.showProfessors();

}

}