**1-- Java Collecjavations**

**2--Threads, Executors, and Runnables**

**3--Application Controller Pattern**

**4--MVC**

**5--Hibernate**

6--QCJSON and JSON

7--Android HttpUrlConnection

8--Servlets

9--JUnit Tests

10--System Level Tests

11--Use Case Diagrams

12--Use Case Documents

13--State Diagrams

14--Sequence Diagrams

==========================================

1. **Java Collections:**

import java.util.ArrayList;

import java.util.Collections;

public class TestSorting {

public ArrayList<Integer> createArray(){

ArrayList<Integer> arr = new ArrayList<Integer>();

arr.add(8);

arr.add(4);

arr.add(3);

arr.add(2);

arr.add(5);

return arr;

}

public void displayArray(ArrayList<Integer> arr){

for(int i=0; i<arr.size(); i++){

System.out.print(arr.get(i)+",");

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

TestSorting obj1 = new TestSorting();

ArrayList<Integer> arr = obj1.createArray();

obj1.displayArray(arr);

Collections.sort(arr);

System.out.println("after sorting");

obj1.displayArray(arr);

}

}

The above code is of java collections, in which an array list of integers is created and a number of integers are added in the array list. The array list is then sorted and displayed

1. **Threads:**
2. **Using Thread class:**

public class TryThread extends Thread {

public TryThread(String firstName, String secondName, long delay) {

this.firstName = firstName; // Store the first name

this.secondName = secondName; // Store the second name

aWhile = delay; // Store the delay

setDaemon(true); // Thread is daemon

}

public static void main(String[] args) {

// Create three threads

Thread first = new TryThread("Hopalong ", "Cassidy ", 200L);

Thread second = new TryThread("Marilyn ", "Monroe ", 300L);

Thread third = new TryThread("Slim ", "Pickens ", 500L);

System.out.println("Press Enter when you have had enough...\n");

first.start(); // Start the first thread

second.start(); // Start the second thread

third.start(); // Start the third thread

try {

System.in.read(); // Wait until Enter key pressed

System.out.println("Enter pressed...\n");

} catch (IOException e) { // Handle IO exception

System.out.println(e); // Output the exception

}

System.out.println("Ending main()");

return;

}

// Method where thread execution will start

public void run() {

try {

while(true) { // Loop indefinitely...

System.out.print(firstName); // Output first name

sleep(aWhile); // Wait aWhile msec.

System.out.print(secondName + "\n"); // Output second name

}

} catch(InterruptedException e) { // Handle thread interruption

System.out.println(firstName + secondName + e); // Output the exception

}

}

private String firstName; // Store for first name

private String secondName; // Store for second name

private long aWhile; // Delay in milliseconds

}

In the above code the thread is created by inheriting the thread class of java, try thread is a user defined class containing first name, last name and delay. Three instances of this thread are created in the main.

1. **Using Runnable:**

import java.io.IOException;

public class JumbleNames implements Runnable {

// Constructor

public JumbleNames(String firstName, String secondName, long delay) {

this.firstName = firstName; // Store the first name

this.secondName = secondName; // Store the second name

aWhile = delay; // Store the delay

}

// Method where thread execution will start

public void run() {

try {

while(true) { // Loop indefinitely...

System.out.print(firstName); // Output first name

Thread.sleep(aWhile); // Wait aWhile msec.

System.out.print(secondName+"\n"); // Output second name

}

} catch(InterruptedException e) { // Handle thread interruption

System.out.println(firstName + secondName + e); // Output the exception

}

}

public static void main(String[] args) {

// Create three threads

Thread first = new Thread(new JumbleNames("Hopalong ", "Cassidy ", 200L));

Thread second = new Thread(new JumbleNames("Marilyn ", "Monroe ", 300L));

Thread third = new Thread(new JumbleNames("Slim ", "Pickens ", 500L));

// Set threads as daemon

first.setDaemon(true);

second.setDaemon(true);

third.setDaemon(true);

System.out.println("Press Enter when you have had enough...\n");

first.start(); // Start the first thread

second.start(); // Start the second thread

third.start(); // Start the third thread

try {

System.in.read(); // Wait until Enter key pressed

System.out.println("Enter pressed...\n");

} catch (IOException e) { // Handle IO exception

System.out.println(e); // Output the exception

}

System.out.println("Ending main()");

return;

}

private String firstName; // Store for first name

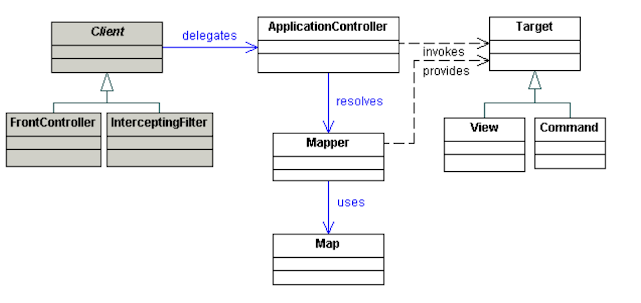
private String secondName; // Store for second name

private long aWhile; // Delay in milliseconds

}

In the above code the thread is created by implementing the runnable interface of java using the second method for thread creation, JumbleNames is a user defined class containing first name, last name and delay. Three instances of this thread are created in the main.

1. **Application Controller Design Pattern:**



The class diagram of Application controller pattern is shown above. The purpose of this pattern is to provide a centralized mechanism for the retrieval and invocation of the request processing components such as commands and views

1. **MVC:**

**//Model**

public class Student {

private String rollNo;

private String name;

public String getRollNo() {

return rollNo;

}

public void setRollNo(String rollNo) {

this.rollNo = rollNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**//View**

public class StudentView {

public void printStudentDetails(String studentName, String studentRollNo){

System.out.println("Student: ");

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

System.out.println("Roll No: " + studentRollNo);

}

}

**//Controller**

public class StudentController {

private Student model;

private StudentView view;

public StudentController(Student model, StudentView view){

this.model = model;

this.view = view;

}

public void setStudentName(String name){

model.setName(name);

}

public String getStudentName(){

return model.getName();

}

public void setStudentRollNo(String rollNo){

model.setRollNo(rollNo);

}

public String getStudentRollNo(){

return model.getRollNo();

}

public void updateView(){

view.printStudentDetails(model.getName(), model.getRollNo());

}

}

In the above code the MVC (Model View and Controller) framework of java has been shown. The code has three independent components, in which the view has the interface to communicate with the end user, the model communicates with the database and the controller gets the requests from views and pass them to controller and vice versa.

1. **Hibernate:**

**//Employee class**

public class Employee {

private int id;

private String firstName, lastName;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

**//StoreData class**

import java.util.List;

import org.hibernate.Query;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class StoreData {

public static void main(String[] args) {

// creating configuration object

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");// populates the data of the // configuration file

// creating seession factory object

SessionFactory factory = cfg.buildSessionFactory();

// creating session object

Session session = factory.openSession();

// creating transaction object

Transaction t = session.beginTransaction();

Employee e1 = new Employee();

//e1.setId(1000);

e1.setFirstName("Irfan");

e1.setLastName("Younas");

session.persist(e1);// persisting the object

Employee emp;

Query q = session.createQuery("FROM com.nu.Employee");

List records = q.list();

emp = (Employee) session.get(Employee.class, new Integer(1));

System.out.println(emp.getFirstName());

t.commit();// transaction is committed

session.close();

System.out.println("successfully saved");

}

}

**//hibernate mapping**

<hibernate-mapping>

<class name="com.nu.Employee" table="employee8">

<id name="id">

<generator class="increment"></generator>

</id>

<property name="firstName"></property>

<property name="lastName"></property>

</class>

</hibernate-mapping>

In the above code the hibernate framework of java is implemented, in which an employee class has been created and store data class is created to store the data and the mapping is done using xml file as shown in hibernate mapping. The employee class contains the id, first and last name of employee.