**ICS 102 Lab13- Arrays of References and Static Members**

**Objectives:**

To introduce students to:

* Linear Search on Arrays of References
* Static members

1. **Linear Searching Arrays of References**

Example:

**public** **class Student {**

**private int id;**

**private String name;**

**private double gpa;**

**public Student(){**

**}**

**public Student(int id, String name, double gpa){**

**this.id = id;**

**this.name = name;**

**this.gpa = gpa;**

**}**

**public int getID(){**

**return id;**

**}**

**public String getName(){**

**return name;**

**}**

**public double getGPA(){**

**return gpa;**

**}**

**public void setID(int id){**

**this.id = id;**

**}**

**public void setName(String name){**

**this.name = name;**

**}**

**public void setGpa(double gpa){**

**this.gpa = gpa;**

**}**

**public String toString(){**

**return "[id = " + id+ ", name = " + name + ", GPA = " + gpa + "]";**

**}**

**public boolean equals(Object obj){**

**if(obj == null)**

**return false;**

**else if(getClass() != obj.getClass())**

**return false;**

**else {**

**Student student = (Student)obj;**

**return this.id == student.id;**

**}**

**}**

**}**

**public class StudentDriver{**

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

**Student[] studentArray = new Student[5];**

**studentArray[0] = new Student(200012, "Farouq Khan",2.0);**

**studentArray[1] = new Student(200010, "Qasim Yusuf",3.2);**

**studentArray[2] = new Student(200015, "Bakari Saleem",4.0);**

**studentArray[3] = new Student(200013, "Ahmad Muhammad",3.5);**

**studentArray[4] = new Student(200011, "Daud Abdullatif",2.5);**

**//Search for a student with ID# 200013:**

**Student student1 = new Student(); // Create a dummy search object**

**student1.setID(200013);**

**boolean found = false;**

**int k;**

**for(k = 0; k < studentArray.length; k++){**

**if(student1.equals(studentArray[k])){**

**found = true;**

**break;**

**}**

**}**

**if(found){**

**System.out.println("Student with id "**

**+ student1.getID() + " found, details are: ");**

**System.out.println(studentArray[k]);**

**}**

**else**

**System.out.println("Student with id "**

**+ student1.getID() + " not found");**

**// Display all students with GPA > 3.0 if any:**

**found = false;**

**for(k = 0; k < studentArray.length; k++){**

**if(studentArray[k].getGPA() > 3.0){**

**found = true;**

**break;**

**}**

**}**

**if(found){**

**System.out.println("\nStudent(s) with GPA > 3.0 are: ");**

**for(int m = k; m < studentArray.length; m++)**

**if(studentArray[m].getGPA() > 3.0)**

**System.out.println(studentArray[m]);**

**}**

**else**

**System.out.println("\nThere are no students with GPA > 3.0");**

**}**

**}**

Alternatively, the two searches can be performed by implementing two search methods:

**public class StudentDriver3{**

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

**Student[] studentArray = new Student[5];**

**studentArray[0] = new Student(200012, "Farouq Khan",2.0);**

**studentArray[1] = new Student(200010, "Qasim Yusuf",3.2);**

**studentArray[2] = new Student(200015, "Bakari Saleem",4.0);**

**studentArray[3] = new Student(200013, "Ahmad Muhammad",3.5);**

**studentArray[4] = new Student(200011, "Daud Abdullatif",2.5);**

**Student student1 = new Student(); // Create a dummy search object**

**student1.setID(200013);**

**int index = linearSearch(studentArray, student1);**

**if(index >= 0){**

**System.out.println("Student with id "**

**+ student1.getID() + " found, details are: ");**

**System.out.println(studentArray[index]);**

**}**

**else**

**System.out.println("Student with id "**

**+ student1.getID() + " not found");**

**gpaGreaterThan3(studentArray);**

**}**

**public static int linearSearch(Student[] studentArray, Student student){**

**for(int k = 0; k < studentArray.length; k++)**

**if(student.equals(studentArray[k]))**

**return k;**

**return -1;**

**}**

**public static void gpaGreaterThan3(Student[] studentArray){**

**int k;**

**for(k = 0; k < studentArray.length; k++){ // find the first student with GPA > 3.0**

**if(studentArray[k].getGPA() > 3.0){**

**break;**

**}**

**}**

**if(k <= studentArray.length - 1){ // if a student with GPA > 3 exists**

**System.out.println("\nStudent(s) with GPA > 3.0 are: "); // display the header**

**// then find the remaining students with GPA > 3 if any**

**for(int m = k; m < studentArray.length; m++)**

**if(studentArray[m].getGPA() > 3.0)**

**System.out.println(studentArray[m]);**

**}**

**else**

**System.out.println("\nThere are no students with GPA > 3.0");**

**}**

**}**

1. **Static members**

static variable:

* It is a variable which belongs to the class and not to an object (an instance). A static variable provides a single copy to be shared by all instances of the class.
* Static variables are initialized only once, at the start of the program execution. These variables will be initialized first, before the initialization of any instance variables.
* A public static variable can be accessed directly, outside its class, by the class name and does not need to be accessed by creating an object:
  + Syntax : **<class-name>.<variable-name>**

static method:

* It is a method which belongs to the class and not to a particular object (a particular instance).

Its definition has the form:

**accessSpecifier static returnType methodName(listOfParameters){**

**statements**

**}**

The return type is **void** if the method does not return a value.

* A static method can access only static data. It cannot access non-static data (i.e., instance variables) directly. It can only do so by an object reference.
* A static method can call only other static methods and cannot call a non-static method directly. It can only call a non-static method by an object reference.
* A public static method can be accessed directly, outside its class, by the class name and does not need to be invoked by creating an object. The method call has the form:

**ClassName.methodName(listOfArguments)**

if the call is in the same class as the static method, then it may be called by:

**methodName(listOfArguments)**

* A static method cannot refer to **this** keyword in anyway.

**Note:**

• Instance methods can access instance variables and instance methods directly provided they are in

scope.

• Instance methods can access class variables and class methods directly provided they are in scope.

Examples:

**double x = Math.sqrt(16.0);**

**System.out.println(Math.PI);**

**boolean flag = Character.isUpperCase('A');**

**String str = String.format("ID: %d, Salary %.2f Saudi Riyals", id, salary);**

**Exercise01:**

|  |
| --- |
| class Student {  int a; //initialized to zero for each object  static int b; //initialized to zero only when class is loaded not for each object created.  Student(){  //Constructor incrementing static variable b  b++;  }  public void showData(){  System.out.println("Value of a = " + a);  System.out.println("Value of b = " + b);  }  //public static void increment(){  //a++;  //}  }  public class Demo{  public static void main(String args[]){  Student s1 = new Student();  s1.showData();  Student s2 = new Student();  s2.showData();  //Student.b++;  //s1.showData();  //s1.increment();  //s1.showData();  //s2.showData();  }  } |

Step1) Compile and run the given Exercise01 Java program

What is the output?

Step 2) It is possible to access a non-private static variable from outside the class using the syntax **ClassName.Variable\_Name**

Uncomment lines 26 and 27:

**//Student.b++;**

**//s1.showData();**

Save , Compile and Run the program. Observe the output.

Step 3) Uncomment lines 14, 15, and 16:

**//public static void increment(){**

**//a++;**

**//}**

Save , Compile and Run the program. Observe the output.

Step 4) Error = ? This is because it is not possible to access instance variable “a” from static method “increment“.

Step 5) Change the method increment() to be an instance method. Uncomment lines 28, 29, and30. Save, Compile and Run the program. Observe the output.

**Exercise02:** Study the following classes that are in different Java files and then determine the output of **TheClassDriver2** and **TheClassDriver1** before executing them:

|  |
| --- |
| public class TheClass{  private static int count;  public TheClass(){  count++;  }  public static int getCount(){  return count;  }  } |

|  |
| --- |
| public class TheClassDriver2 {  public static void main(String[] args) {  System.out.println("One TheClass object created in the main of TheClassDriver2:");  TheClass obj2 = new TheClass();  System.out.println("count = " + TheClass.getCount());    mymethod();    System.out.println("Total number of created TheClass objects = " + TheClass.getCount());  }    public static void mymethod(){  System.out.println("Two TheClass objects created in mymethod of TheClassDriver2:");  TheClass obj1 = new TheClass();  TheClass obj2 = new TheClass();  System.out.println("count = " + TheClass.getCount());  System.out.println("One more TheClass object created in mymethod of TheClassDriver2:");  TheClass obj3 = new TheClass();  System.out.println("count = " + TheClass.getCount());  }  } |
| public class TheClassDriver1 {  public static void main(String[] args) {  System.out.println("Two TheClass objects created in main of TheClassDriver1:");  TheClass obj1 = new TheClass();  System.out.println("count = " + TheClass.getCount());  TheClass obj2 = new TheClass();  System.out.println("count = " + TheClass.getCount());    TheClassDriver2.mymethod();    System.out.println("Last TheClass object created in main of TheClassDriver1:");  TheClass obj4 = new TheClass();  System.out.println("count = " + TheClass.getCount());  System.out.println("Total number of created TheClass objects = " + TheClass.getCount());  }  } |

**Laboratory Tasks**

**Task 01:**

(a)Write a class **Fruit** with:

* private instance variables name, and pricePerKilogram
* Appropriate constructor
* Appropriate accessor methods
* Appropriate mutator methods
* toString method
* equals method

(b) Write a **FruitDriver** class that:

* Initializes an array of **Fruit** objects, **fruitArray**, with 10 objects with names: banana, apple, mango, orange,pineapple, pear, grapes, tangerine, watermelon, sweetmelon and appropriate prices per kilogram.
* Uses an appropriate loop to display all objects with **pricePerKilogram > 5.00** Saudi Riyals, if any.
* Calls a linearSearch method:

**public static boolean findFruit(Fruit targetFruit, Fruit[] fruitArray)**

the method ruturns ***true*** if **targetFruit** is in the **fruitArray**; otherwise it returns ***false***.

* Displays the result of the search using an appropriate message

**Task02:** A company operates several sandwitch stands in Al-Khobar. Complete the class **SandwitchStand** and a **SandwitchStandDriver** class such that it can be used to display the total number of sandwitches sold in each created **SandwitchStand** object and the total number of sandwitches sold in all sandwitch stands.

Sample program output is:

|  |
| --- |
| **Total number of sandwitches sold before object creation = 0**  **Total number of sandwitches sold at stand 1234 before any sells = 0**  **Total number of sandwitches sold at stand 1234 = 2**  **Total number of sandwitches sold at stand 5678 = 3**  **Total number of sandwitches sold at stand 5234 = 1**  **Total number of sandwitches sold in all stands = 6** |

The class SandwitchStand has:

* Variables:
  + private static int totalSandwitchSoldAllStands
  + private int totalSandwitchSoldAtStand
  + private int standID
* Constructor:
  + SandwitchStand(int standID)
* Methods:
  + public int getStandID()
  + public int getNumSandwitchSoldAtStand()
  + public void sellSandwitch(int numberOfSandwiches) throws IllegalArgumentException
  + public static int getNumSandwitchSoldAllStands()

**Task 03:** Given the following class:

**public class Point {**

**private double x;**

**private double y;**

**public Point(double x, double y){**

**this.x = x;**

**this.y = y;**

**}**

**public double getX(){**

**return x;**

**}**

**public double getY(){**

**return y;**

**}**

**public double distanceFrom(Point other){**

**return Math.hypot(this.x - other.x , this.y - other.y);**

**}**

**}**

Write a driver program that:

* Creates an array of type **Point** with 5 point objects. Use these points:

(3, 5) (3, 8) (4, -1) (2, -3) (1, 7)

* Finds the point in the array that is the closest to the origin (0, 0) and print it out [Note: Provide a **toString( )** method for this]

**Task 04**: Write a Java application that initializes an array of Strings with the following strings:

"Hot Peppers" "Onions" "Milk" "Eggs" "Rice"

The program must then use a loop to get the first character of each string in the array and make a new word with those letters. The program finally prints the new word.