01)

Source Code:

StudentList class:

```
import java.util.Arrays;

public class StudentList {
    private int maxSize ;
    private int position;
    private Student[] ListEntry;

    public StudentList(int size){
        maxSize = size;
        ListEntry = new Student[maxSize];
        position= -1;
    }

    public boolean IsListEmpty(){
        return (position==-1);
    }

    public boolean IsListFull(){
        return (position== maxSize-1);
    }

    public int ListSize(){
        return (position+1);
    }

    public void InsertLast(Student student){
        if (IsListFull())
            System.out.println("Attempt to insert at the end of a full
list");
        else
            ListEntry[++position] = student;
    }

    public void InsertList(int p, Student student){
        int i;
        if (IsListFull())
            System.out.println("Attempt to insert an entry into a full
list");
        else if (p < 0 || p > ListSize())
            System.out.println("attempt to insert a position not in the
list");
        else
        {
            for( i = ListSize(); i >p; i--)
                ListEntry[i] = ListEntry[i-1];
            ListEntry[p] = student;
            position++;
        }
    }
    public Student RetrieveList(int index){
        Student student;
        if (IsListEmpty()){
            System.out.println("List is empty");
            return null;
        } else if(index < 0 || index >= ListSize()){
            System.out.println("Out of list size. Enter a valid index.");
            return null;
        } else {
            student = ListEntry[index];
            return student;
        }
    }


    public void DeleteList( int p) {
        int i;
        Student student;
        if (IsListEmpty())
            System.out.println("Attempt to delete an entry from an empty
list");
        else if (p < 0 || p >= ListSize())
            System.out.println("attempt to delete a position not in the
list");
        else {
            student = ListEntry[p];
            for( i = p; i < ListSize()-1; i++)
                ListEntry[i] = ListEntry[i+1];
            position--;
        }
    }

    public void ReplaceList (int p, Student student){
        int i;
        if (IsListEmpty())
            System.out.println("Attempt to replace an entry of an empty
list");
        else if (p < 0 || p >= ListSize())
            System.out.println("attempt to replace an entry at a position not
in the list");
        else
            ListEntry[p] = student;
    }

    public void TraverselList(){
        int i;
        for (i=0; i<position+1; i++)
            System.out.println(ListEntry[i]);
    }

}
```

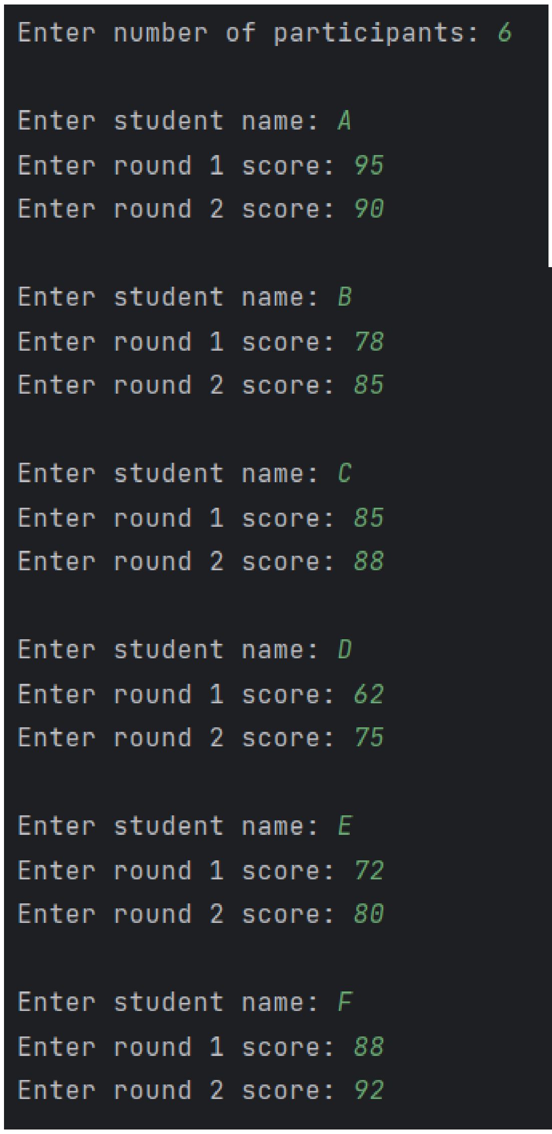
Student class:

```java
public class Student {
    private final String name;
    private final int round1Marks;
    private final int round2Marks;

    public Student(String name, int round1Marks, int round2Marks) {
        this.name = name;
        this.round1Marks = round1Marks;
        this.round2Marks = round2Marks;
    }

    public String getName() {
        return name;
    }

    public int getRound1Marks() {
        return round1Marks;
    }

    public int getRound2Marks() {
        return round2Marks;
    }

}
```

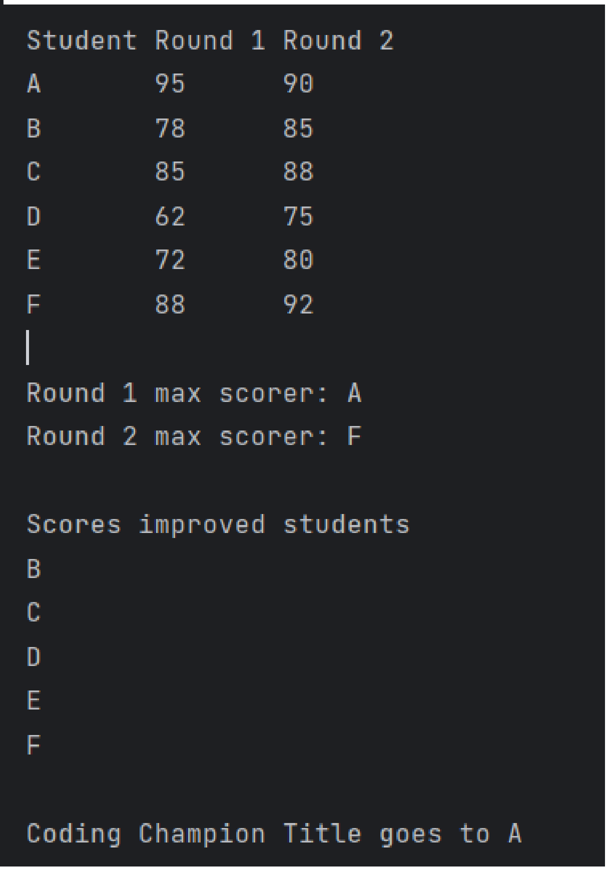
UOKCoding class:

```java
import java.util.Scanner;

public class UOKCoding {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of participants: ");
        int numStudent = scanner.nextInt();
        System.out.println();

        StudentList list = new StudentList(numStudent);

        // get participant details
        for (int i = 0; i < numStudent; i++) {
            System.out.print("Enter student name: ");
            String name = scanner.next();

            System.out.print("Enter round 1 score: ");
            int round1Marks = scanner.nextInt();

            System.out.print("Enter round 2 score: ");
            int round2Marks = scanner.nextInt();
            System.out.println();

            Student student = new Student(name, round1Marks, round2Marks);
            list.InsertLast(student);
        }
        // list all participants
        System.out.println();
        System.out.println("Student\tRound 1\tRound 2");
        for (int i = 0; i < numStudent; i++) {
            Student student = list.RetrieveList(i);
            System.out.println(student.getName() + "\t\t" +
student.getRound1Marks() + "\t\t" + student.getRound2Marks());
        }
        System.out.println();

        // find max scorer
        int maxRound1 = 0, marksRound1, maxRound2 = 0, marksRound2,
winnerScore = 0, totScore;
        String maxScorerRound1 = "", maxScorerRound2 = "", winner = "";
        for (int i = 0; i < numStudent; i++) {
            Student student = list.RetrieveList(i);
            marksRound1 = student.getRound1Marks();
            marksRound2 = student.getRound2Marks();
            totScore = student.getRound1Marks() + student.getRound2Marks();

            if (marksRound1 > maxRound1){
                maxRound1 = marksRound1;
                maxScorerRound1 = student.getName();
            }
            if (marksRound2 > maxRound2){
                maxRound2 = marksRound2;
                maxScorerRound2 = student.getName();
            }

            if (totScore > winnerScore){
                winnerScore = totScore;
                winner = student.getName();
            }
        }
        System.out.println("Round 1 max scorer: " + maxScorerRound1);
        System.out.println("Round 2 max scorer: " + maxScorerRound2);
        System.out.println();

        displayImprovedScores(list, numStudent);
        System.out.println();

        System.out.println("Coding Champion Title goes to " + winner);
    }

    public static void displayImprovedScores(StudentList list, int
numStudent){
        System.out.println("Scores improved students");
        for (int i = 0; i < numStudent; i++) {
            Student student = list.RetrieveList(i);
            if (student.getRound1Marks() < student.getRound2Marks()){
                System.out.println(student.getName());
            }
        }
    }
}
```

Output:



```
Enter number of participants: 6

Enter student name: A
Enter round 1 score: 95
Enter round 2 score: 90

Enter student name: B
Enter round 1 score: 78
Enter round 2 score: 85

Enter student name: C
Enter round 1 score: 85
Enter round 2 score: 88

Enter student name: D
Enter round 1 score: 62
Enter round 2 score: 75

Enter student name: E
Enter round 1 score: 72
Enter round 2 score: 80

Enter student name: F
Enter round 1 score: 88
Enter round 2 score: 92
```



```
Student Round 1 Round 2
A       95      90
B       78      85
C       85      88
D       62      75
E       72      80
F       88      92

Round 1 max scorer: A
Round 2 max scorer: F

Scores improved students
B
C
D
E
F

Coding Champion Title goes to A
```

02)

Source Code:

List class:

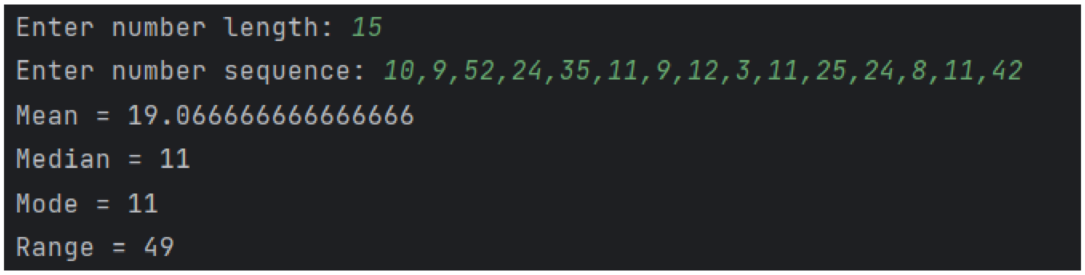
```
import java.util.Arrays;

public class List {
    private int maxSize ;
    private int position;
    private int[] ListEntry;

    public List(int size){
        maxSize = size;
        ListEntry = new int[maxSize];
        position= -1;
    }

    public boolean IsListEmpty(){
        return (position==-1);
    }

    public boolean IsListFull(){
        return (position== maxSize-1);
    }

    public int ListSize(){
        return (position+1);
    }

    public void InsertLast(int x){
        if (IsListFull())
            System.out.println("Attempt to insert at the end of a full
list");
        else
            ListEntry[++position] = x;
    }

    public void InsertList(int p, int element){
        int i;
        if (IsListFull())
            System.out.println("Attempt to insert an entry into a full
list");
        else if (p < 0 || p > ListSize())
            System.out.println("attempt to insert a position not in the
list");
        else
        {
            for( i = ListSize(); i >p; i--)
                ListEntry[i] = ListEntry[i-1];
            ListEntry[p] = element;
            position++;
        }
    }
    public int DeleteList( int p)  {
        int i,element;
        if (IsListEmpty())
            System.out.println("Attempt to delete an entry from an empty
list");
        else if (p < 0 || p >= ListSize())
            System.out.println("attempt to delete a position not in the
list");
        else {
            element = ListEntry[p];
            for( i = p; i < ListSize()-1; i++)
                ListEntry[i] = ListEntry[i+1];
            position--;
            return element;
        }
        return 0;
    }

    int RetrieveList(int p ){
        int i,element;
        if (IsListEmpty()){
            System.out.println("Attempt to retrieve an entry from an empty
list");
            return 0;}
        else if (p < 0 || p >= ListSize()){
            System.out.println("attempt to retrieve an entry at a position
not in the list");
            return 0; }
        else{
            element = ListEntry[p];
            return element;}
    }

    public void ReplaceList (int p, int x){
        int i;
        if (IsListEmpty())
            System.out.println("Attempt to replace an entry of an empty
list");
        else if (p < 0 || p >= ListSize())
            System.out.println("attempt to replace an entry at a position not
in the list");
        else
            ListEntry[p] = x;
    }

    public void TraverselList(){
        int i;
        for (i=0; i<position+1; i++)
            System.out.println(ListEntry[i]);
    }

    public void sortList(){
        Arrays.sort(ListEntry);
    }
}
```

MeanMedianModeRange class:

```
import java.util.Scanner;

public class MeanMedianModeRange {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number length: ");
        int len = scanner.nextInt();

        System.out.print("Enter number sequence: ");
        String input = scanner.next();

        String[] numberStrings = input.split(",");
        List listNumber = new List(len);
        for (int i = 0; i < len; i++) {
            listNumber.InsertLast(Integer.parseInt(numberStrings[i]));
        }
        checkMean(listNumber);
        checkMedian(listNumber);
        checkMode(listNumber);
        checkRange(listNumber);
    }

    public static void checkMean(List list){
        double total = 0, mean;
        for (int i = 0; i <= list.ListSize()-1; i++) {
            total += list.RetrieveList(i);
        }
        mean = total/list.ListSize();
        System.out.println("Mean = " + mean);
    }

    public static void checkMedian(List list){
        double median;
        int midPosition;
        list.sortList();
        if (list.ListSize()%2 == 1){
            midPosition = (list.ListSize() + 1)/2;
            median = list.RetrieveList(--midPosition);
            System.out.println("Median = " + (int) median);
        } else if (list.ListSize()%2 == 0) {
            int prevPosition = list.ListSize()/2 - 1;
            int nextPosition = prevPosition++;
            median = (double) (list.RetrieveList(prevPosition) +
list.RetrieveList(nextPosition)) / 2;
            System.out.println("Median = " + median);
        }
    }

    public static void checkMode(List list){
        int mode = list.RetrieveList(0);
        int maxCount = 1;
        for (int i = 0; i < list.ListSize(); i++) {
            int count = 0;
            for (int j = 0; j < list.ListSize(); j++) {
                if (list.RetrieveList(i) == list.RetrieveList(j)){
                    count++;
                }
            }
            if (count > maxCount){
                maxCount = count;
                mode = list.RetrieveList(i);
            }
        }
        System.out.println("Mode = " + mode);
    }

    public static void checkRange(List list){
        list.sortList();
        System.out.println("Range = " + (list.RetrieveList(list.ListSize() -
1) - list.RetrieveList(0)));
```

```
    }
}
```

Output:



```
Enter number length: 15
Enter number sequence: 10,9,52,24,35,11,9,12,3,11,25,24,8,11,42
Mean = 19.066666666666666
Median = 11
Mode = 11
Range = 49
```