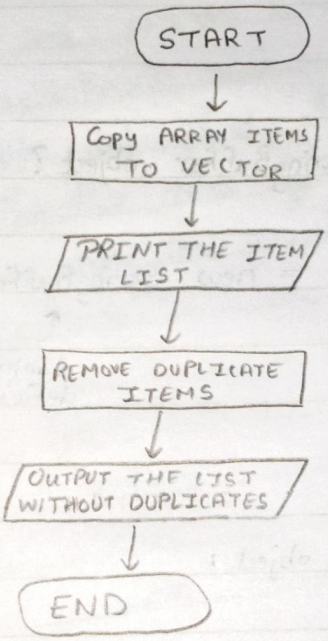
Practical No. 05

Aim : Create, debug and run java programs based on Wrapper class and Vectors

Flow chart :



code 1 : VECTOR

Practical No. 05

Aim: Create, debug and ~~rerun~~ java programs based on Wrapper class and vectors.

Theory:

What are ~~wrapper~~ classes?

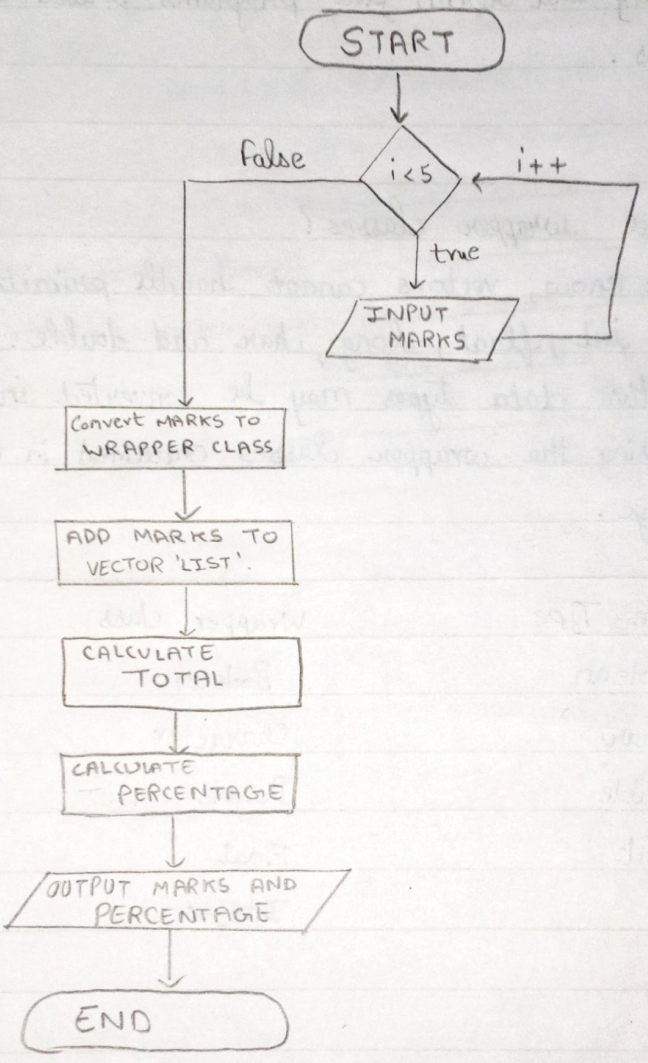
→ • As we know, vectors cannot handle primitive data types like int, float, long, char and double.
• Primitive data types may be converted into object types by using the wrapper classes contained in the java.lang package.

Simple Type	Wrapper class
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer

Autoboxing:

• The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing.
• Since Java 5, we do not need to use the ~~valueOf()~~ valueOf() method.
• example.

```
int a = 25;
Integer b = ~~new Integer(a);~~ Integer.valueOf(a);
```



Code 2: Wrapper Class

Unboxing:

- Automatic conversion of wrapper class type into its corresponding primitive type is known as unboxing.
- This process is reverse of autoboxing.
- Example:

```
Integer a = new Integer(35);
int b = a; // now, since Java 5, compiler will write
           //a.intValue(); internally.
```

Vectors in Java:

- The Vector class in Java implements a growable array of objects.
- To use Vector class, we need to import 'java.util.Vector'.
- Vector implements a dynamic array, i.e. it can grow and shrink size according to the need.

Declaration:

```
Vector<e> v = new Vector<e>();
ex. Vector<String> v = new Vector<String>();
```

Vector methods:

```
v.addElement(item);
v.elementAt(10);
v.size();
v.removeElement(item);
v.removeElementAt(index);
```

Conclusion :

Hence, by performing this practical I learnt about the concepts of Wrapper classes and Vector. I also coded, debugged and executed java programs based on the concepts of Wrapper classes and Vectors.