Topics

- Vector class in Java

Vector class in Java

- Vector class supports growable array of objects.
- Two Types of Vectors : Parameterized (Can Hold Values of only one Type) and Un-parameterized(can Hold Values of Various Types)
- Constructors

❑ Vector() [Un-parameterized],
Vector<T>() [Parameterized]
size() = 0 , capacity = 10 , increment = 2*capacity

❑ Vector(int size) [Un-parameterized],
Vector<T>(int size) [Parameterized]
size() = 0 , capacity = size, increment = 2*capacity

❑ Vector(int size, int incr) [Unparameterized],
Vector<T>(int size, int incr) [Parameterized]
size() =0, capacity = size, increment = incr

Vector vs Array

1. Array holds elements of only one type. Vector can hold elements of various types [However, parameterized vectors are preferred]
2. Array elements are referred via subscripts such as data[0], marks[2][3]. Vector elements are also indexed but are manipulated via add(..), get(..), set(..) methods. For example, suppose 'v' is a vector instance, then v.get(0) returns element stored at index 0.
3. Array can be 1-D, 2-D etc. However there is no dimension in Vector.
4. Arrays have <<length>> as an attribute. Vector uses size() method to get its current size .
5. ArrayIndexOutofBoundsException (if an element is referred out of array index) occurs in Arrays. In Vector the corresponding exception is IndexOutofBoundsException.

Important Methods of Vector class

- int size() → Returns the size of Vector
- int capacity() → Returns the capcity of Vector
- boolean add(E e) → Adds ‘e’ of type ‘E’ at end of vector [Parameterized]
- boolean add(Object o) → Adds ‘o’ of type Object at end of vector [Un-Parameterized]
- boolean add(int index, E e) → Adds ‘e’ of type ‘E’ at a specified index (0<=index<=size-of-vector) [Parameterized]
- Booean add(int index, Object o) → Adds ‘o’ of type ‘Object’ at a specified index (0<=index<=size-of-vector) [Un-Parameterized]
- boolean addAll(Collection c) → Adds the elements of collection ‘c’ at the end of the invoking collection
- boolean addAll(int index, Collection c) → Adds the elements of collection c in the current vector starting from index
- E get(int index) → Returns the element of type ‘E’ from index. Throws IndexOutofBoundsException if index >= size. [Parameterized]
- Object get(int index) → Returns the element of type ‘Object’ from index. Throws IndexOutofBoundsException if index >= size. [Un-Parameterized]
- void clear() → Removes all the elements and sets size = 0
- void remove (int index) → Removes an elements from index (0<=index<size)
- void remove (Object o) → Removes the first occurrence of ‘o’ if exists otherwise no effect
- void insertElement(int index, Object o) → Inserts an element ‘o’ at index (0<=index<size).
- void set(int index, E e) → Sets the element at index via element ‘e’ [Update/modify the previous value]

Vector class : Example 1 ….

```
import java.util.*; ·····················→ To use Vector class
class VectorDemo
{
        public static void main(String args[])
        {
                Vector v1 = new Vector(5); ·····→ Un-parameterized Vector
                System.out.println(v1.size()); ·········→ 0
                System.out.println(v1.capacity()); ·····→ 5

                v1.add(10); ·····················→ Adds 10 (int) at index 0

                v1.add(20.5); ···················→ Adds 20.5 (double) at index 1

                v1.add(3.6f); ···················→ Adds 3.6f (float) at index 2

                v1.add("Object"); ···············→ Adds "Object" (String) at index 3

                v1.add(5); ······················→ Adds 5 (int) at index 4

                v1.add(6); ······················→ Adds 6 (int) at index 5

                System.out.println(v1.size()); ·········→ 6
                System.out.println(v1.capacity()); ·····→ 10
        }// End of Method
}// End of VectorDemo class
```

Vector class : Example 1

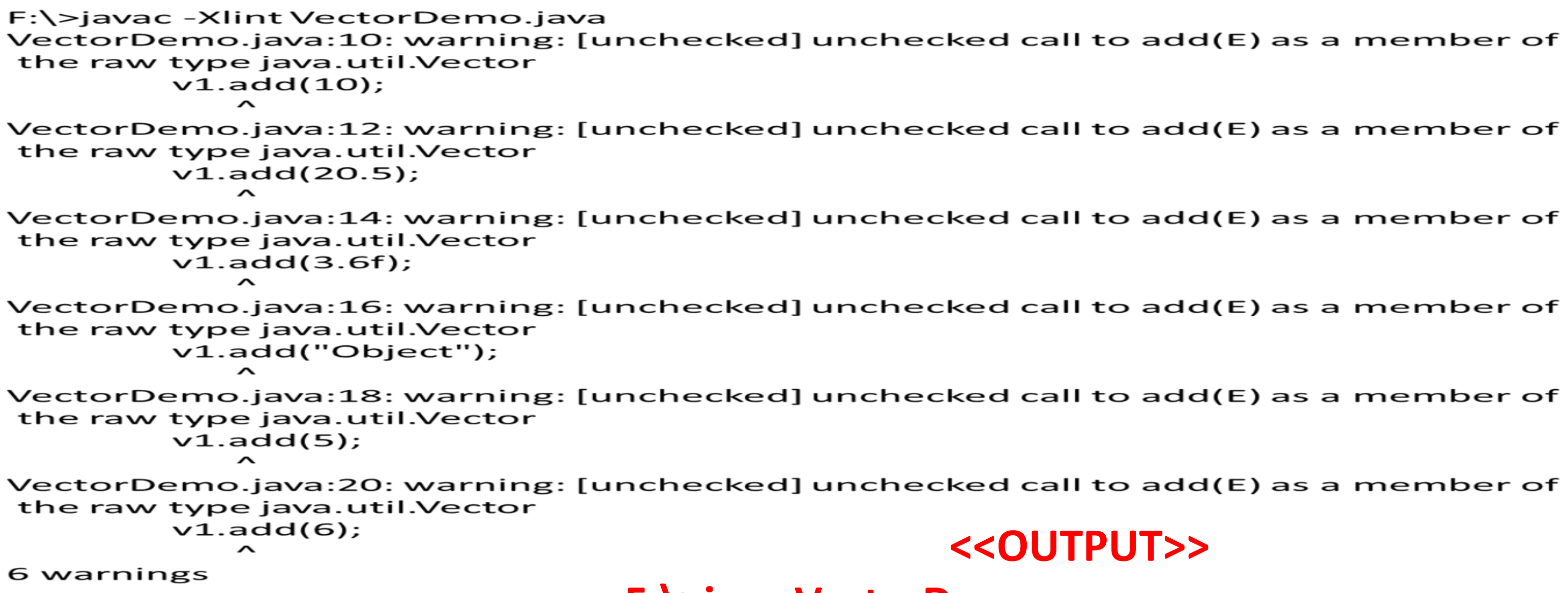
- If you are using an Un-parameterized Vectors in your program then you have to compile the program using <<–Xlint>> option
- Syntax : javac   -Xlint   name-of-source-file

Error if compiled w/o –Xlint

```
F:\>javac VectorDemo.java
Note: VectorDemo.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Vector class : Example 1 ….

- **Compiling with –Xlint option will results in warnings not errors.**



```
F:\>javac -Xlint VectorDemo.java
VectorDemo.java:10: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add(10);
                   ^
VectorDemo.java:12: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add(20.5);
                   ^
VectorDemo.java:14: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add(3.6f);
                   ^
VectorDemo.java:16: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add("Object");
                   ^
VectorDemo.java:18: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add(5);
                   ^
VectorDemo.java:20: warning: [unchecked] unchecked call to add(E) as a member of
 the raw type java.util.Vector
             v1.add(6);
                   ^
6 warnings
```

<<OUTPUT>>

```
F:\>java VectorDemo
0
5
6
10
```

Some Facts About Un-parameterized Vectors

- Elements of any type are added and retrieved only in ‘Object’ type.
- You have to type cast the element to its base type before use.
- Example : Sum of numbers stored in vector

```
import java.util.*;
class VectorDemo
{
        public static void main(String args[])
        {
                Vector v1 = new Vector(5);

                v1.add(10); v1.add(20); v1.add(30);

                double sum = 0;
                for(int i =0; i < v1.size(); i++)
                        sum = sum + v1.get(i);

        }// End of Method
}// End of VectorDemo class
```

```
F:\>javac -Xlint VectorDemo.java
VectorDemo.java:8: warning: [unchecked] unchecked call to
add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
              ^
VectorDemo.java:8: warning: [unchecked] unchecked call to
add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
                          ^
VectorDemo.java:8: warning: [unchecked] unchecked call to
add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
                                      ^
VectorDemo.java:12: operator + cannot be applied to
double,java.lang.Object
            sum = sum + v1.get(i);
                      ^
1 error
3 warnings
```

Some Facts About Un-parameterized Vectors ….

```
import java.util.*;
class VectorDemo
{
        public static void main(String args[])
        {
                Vector v1 = new Vector(5);

                v1.add(10); v1.add(20); v1.add(30);

                double sum = 0;
                for(int i =0; i < v1.size(); i++)
                        sum = sum + (Integer) v1.get(i);
                System.out.println(“Sum=“+sum);
        }// End of Method
}// End of VectorDemo class
```

```
F:\>javac -Xlint VectorDemo.java
VectorDemo.java:8: warning: [unchecked] unchecked
call to add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
              ^
VectorDemo.java:8: warning: [unchecked] unchecked
call to add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
                          ^
VectorDemo.java:8: warning: [unchecked] unchecked
call to add(E) as a member of
the raw type java.util.Vector
        v1.add(10); v1.add(20); v1.add(30);
                                      ^
3 warnings

F:\>java VectorDemo
Sum= 60.0
```

Parameterized Vectors

- Holds Elements of only One Type.
- No need to compile using <<-Xlint>> option
- Compile-Time Error →if any other type element is being added

Parameterized Vector of String type

```
import java.util.*;
class VectorDemo
{
		public static void main(String args[])
		{
			Vector<String> v1 = new Vector<String>(5);
			v1.add("10");		//		Correct
			v1.add(20);		//		Compile-Time Error
			v1.add(30);		//		Compile-Time Error

		}// End of Method
}// End of VectorDemo class
```

```
F:\>javac VectorDemo.java
VectorDemo.java:9: cannot find symbol
symbol  : method add(int)
location: class java.util.Vector<java.lang.String>
          v1.add(20);
            ^
VectorDemo.java:10: cannot find symbol
symbol  : method add(int)
location: class java.util.Vector<java.lang.String>
          v1.add(30);
            ^
2 errors
```

innovate achieve lead

Thank You