Assertions

- Introduction
- Assert as keyword and identifier
- Types of assert statement
- Various Runtime flags
- Appropriate and Inappropriate use of assertions
- Assertion Errors

Assertions

Introduction:

1. Very common way of debugging is using System.out.println() statement, but the problem with System.out.println() statement is after fixing the problem we should delete these System.out.println() statements otherwise these statements will be executed at runtime and effect performance and disturb login.

2. To resolve this problem SUN people introduced **assertion concept in 1.4** version, hence the main objective of assertions is to perform debugging.

3. The main advantage of assertions over System.out.println() statement is after fixing the problem it is not required to delete assert statements because assertions will be disabled automatically at runtime. Based on our requirement we can enable and disable assert statements and by **default assertions are disabled.**

4. Assertions concept is applicable for development and test environment but not for production environment.

Assert as keyword and Identifier

1. Assert keyword is introduced in 1.4 version, so we can't use assert as identifier, but before that we can use assert as identifier.

Example:

```
class test
{
public static void main(String ar[ ])
{
int assert=10;
System.out.println(assert)
}
```

```
}
Run:
  Javac test.java
C.E. as of release 1.4 ,'assert' is a keyword and may not be used as an identifier.
Use –source1.3 or lower to use 'assert' as an identifier

 Javac –source1.3 test.java
Java test↵
10 ↵
```

Types of assert statement

There are two types of assert statements

1) Simple Statement

2) Augmental version

1) Simple Statement:

assert (b); b-should be Boolean type

1. If b is true, then our assumption satisfied and rest of the program will be executed normally.

2. If b is false ,then our assumption fails the program will be terminated by raising runtime exception saying assertion error so,that we can able to the fix the problem.

```
Example:
class test
{
public static void main(String ar[])
{
int x=10;
-
-
```

```
-
assert(x>10)
-
-
-
System.out.println(x);
}
}
Run:
1)Javac test.java
2)Javac test
10
3)Javac –ea test
R.E.AssertionError
```

2)Augmented Version:

1. We can Augment description by using augmented version to the assertion error.

assert(b);d ⟶ Any description can be any type but Recommended to use string type

Should be Boolean type

Example:

```
class test
{
        public static void main(String ar[])
        {
        int x=10;

         -

         -

      assert(x>10);     Here x value should be >10 but it is not
```

```
-

-

 System.out.println(x);
}
}
```

Run:
1)Javac test.java
2)Javac test
10
3)Javac –ea test
R.E.AssertionError:Here x value should be >10 but it is not

Conclusion:

assert(e1):e2;

E2 will be evaluated iff e1 is false. i.e. if e1 is true,then e2 won't be evaluated.

Example:

```
class test
{public static void main(String ar[])
{
int x=10;
-
-
-
assert(x==10):++x        assert(x.10):++x;
-
-
System.out.println(x);
}

}
```

```
Run:
1)Javac test.java
2)Javac test
10
3)Javac –ea test
 10
```

Conclusion:

assert(e1):e2:

1. As e2 can take as method call also but void type method calls are not allowed .

Example:

```
class test
{
public static void main(String ar[])
{
int x=10;
-
-
-
assert(x>10):m1        assert(x.10):++x;
-
-
System.out.println(x);
}
Public static int m1()
{
return 8888;
}

}
Run:
1)Javac test.java
```

2)Java test
10
3)Javac –ea test
R.E.AssertionError:8888