Programming in Java

Topic: Assertion

Contents

Introduction

- An assertion is a statement in Java that enables you to test your assumptions about the program.
- Each assertion contains a boolean expression that you believe will be true when the assertion executes.
- If it is not true, the system will throw a runtime error (java.lang.AssertionError).

Example

For example, if we write a method that calculates the interest of any account, we might assert that the roi or amount is positive.

```
double calculateInterest(int amt, double roi, int years)
{
        //Using assertion
        assert amt>0;
        double interest = (amt*roi*years)/100;
        return interest;
}
```

Assertion

- The assertion statement has two forms.
  - assert *Expression1*;

    Here, Expression1 is a boolean expression. When the system runs the assertion, it evaluates Expression1 and if it is false throws an AssertionError with no detail message.

  - assert *Expression1* : *Expression2*;

    Here, Expression1 is a boolean expression and Expression2 is an expression that has a value and it cannot be an invocation of a method that is declared void. If Expression1 fails then Expression2 result will be displayed.

Example


```
public class pqr {
public static void main(String[] args) {
// get a number in the first argument
int num = Integer.parseInt(args[0]);
//assert num <= 10;// stops if number > 10
assert num <=10 : "Number is greater than 10";
System.out.println("All assumptions are correct");
}
}
```

Output: Suppose we have passed 11, then output will be:

```
Exception in thread "main" java.lang.AssertionError: Number is greater than
    10
        at pqr.main(pqr.java:6)
```

On passing value less than or equal to 10, output will be:

All assumptions are correct

Using Assertion

Assertion will not work directly because assertion is disabled by default.

To enable the assertion, -ea or -enableassertions switch of java must be used.

Compile: javac AssertionExample.java

Run: java -ea AssertionExample

How Assertion Works?

In assertion a BooleanExpression is used and if boolean expression is false then java throws AssertionError and the program terminates.

Second form of assert keyword "assert booleanExpression : errorMessage" is more useful and provides a mechanism to pass additional data when Java assertion fails and java throws AssertionError.

Need of Assertion


```
class pqr{
void remainder(int i)
{
if(i%3==0)
System.out.println("Divisible by 3");
else if(i%3==1)
System.out.println("Remainder 1");
else
{
System.out.println("Remainder 2");
}
}
public static void main(String [] args)

{
new pqr().remainder(8);
}
}
```

Output:

Remainder 2

What About it?

```
public class pqr
   {
        public static void main(String [] args)
            {
                    new pqr().remainder(-8);
            }
   }
```

Need of Assertion


```
class pqr{
void remainder(int i)
{
if(i%3==0)
System.out.println("Divisible by 3");
else if(i%3==1)
System.out.println("Remainder 1");
else
{
System.out.println("Remainder 2");
}
}
public static void main(String [] args)
{
new pqr().remainder(-8);
}
}
```

Here output will be:

Remainder 2, which is not correct, it must be Remainder -2, so assertion could be used[Described in next slide]

Previous example using assertion


```
class pqr{
void remainder(int i)
{
if(i%3==0)
System.out.println("Divisible by 3");
else if(i%3==1)
System.out.println("Remainder 1");
else
{
assert i%3==2:"Remainder is negative, not matching with any case";
System.out.println("Remainder 2");
}
}
public static void main(String [] args)
{
new pqr().remainder(-8);
}
}
```

Output:

```
Exception in thread "main" java.lang.AssertionError: Remainder is negative, not matching with any case
        at pqr.remainder(pqr.java:10)
        at pqr.main(pqr.java:16)
```

Why Assertion?

- Using assertion helps to detect bug early in development cycle which is very cost effective.
- Assertion in Java makes debugging easier as AssertionError is a best kind of error while debugging because its clearly tell source and reason of error.
- Assertion is great for data validation if by any chance your function is getting incorrect data your program will fail with AssertionError.

Important

- Assertion is introduced in JDK 1.4 and implemented using assert keyword in java.
- Assertion can be enabled at run time by using switch –ea or –enableassertion
- Assertion does not replace Exception but compliments it.
- Do not use assertion to validate arguments or parameters of public method.(It would be better to throw exceptions rather than using assertion for checking parameter values, as assertions could be disabled also, which can compromise the parameter check sometimes)

Q1

What will happen when you compile and run the following code with assertion enabled?

```
public class Test{
        public static void main(String[] args){

                int age = 20;
                assert age > 20 : getMessage();
                System.out.println("Valid");

        }
        private static void getMessage() {
                System.out.println("Not valid");
        }
}
```

A. The code will not compile

B. The code will compile but will throw AssertionError when executed

C. The code will compile and print Not Valid

D. The code will compile and print Valid

Q2

What will happen when you compile and run the following code with assertion enabled?

```
public class Test{

        public static void main(String[] args){
                assert false;
                System.out.println("True");

        }
}
```

A. The code will not compile due to unreachable code since false is hard coded in assert statement

B. The code will compile but will throw AssertionError when executed

C. The code will compile and print true

D. None of the above

Q3

Output?

```
public class Test{

        public static void main(String[] args){

                boolean b = false;
                assert b = true;
          System.out.println("Hi");
        }
}
```

A. Hi
B. The code will not compile
C. The code will compile but will throw AssertionError when executed
D. None of these

Q4(Output??)

```
public class pqr{
        public static void main(String[] args){
                int i = 10;
                int j = 24;
                int k = 34;
                assert i + j >= k : k--;
                System.out.println(i + j + k);
        }

}
```

A. 67
B. 68
C. Compiles fine but gives AssertionError
D. None of these