Chapter 2

Comments, Identifiers, Keywords

- In this chapter, You will learn
  - The need of comment, types of comments, syntax rules
  - The identifier, rules in defining identifier.
  - Keywords, types of keywords, its rules.
- By the end of this chapter- you can indentify valid and invalid comments, identifiers, and keywords and also all operations we are performing in a Java application.

i

Interview Questions

By the end of this chapter you answer all below interview questions

Java Comments

- Definition of comment
- Why comment?
- Types of comments
- CE: unclosed comment
- CE: class or enum or interface expected
- CE: illegal start of type
- CE: illegal start of expression
- SCJP question

Identifiers

- Definition of identifier
- CE: identifier expected
- Rules in defining Identifier?
- Can we use keyword as an identifier?
- Can we use predefined class name as identifier?
- Why can't you use keyword as identifier for class, method, or variable?
- How can we differentiate user defined and predefined classes if both are defined with same name?
- What is the limitation of identifier length?
- SCJP question

Keywords

- Definition of keyword
- Keyword rules?
- What is the basic use of keyword?
- The list of operations we do in a Java file and class?
- List of keywords
- Is null a keyword?
- SCJP question

iii

Java Comments

Definition

A description about a basic programming element is called comment.

Why comment?

Comments are meant for developer to understand the purpose.

Types of comments

Java supports 3 types of comments

1. Single Line comment - //
2. Multiline comment - /* */
3. Document comment - /** */

The statements placed inside these special characters are ignored by compiler while compiling the class. It means, compiler will not generate bytecodes for those comments. So comments are not appeared in .class file.

Below program shows usage of all above 3 comments

```
//Addition.java
/**
 * This program shows adding two integer numbers.
 * @author: Hari Krishna
 * @company: Naresh i Technologies
 * @Date: 04/13/2011
 * @Version: 1.0
 */

public class Addition
{
        /*
        This method takes two integer numbers and
                return their addition result.
        */

        public static void add(int a, int b)
        {
                //adding and returning result
                return a + b ;
        }
}
```

Find out valid comments from the below list. Valid comments means compiler should not throw compilation time error when it is appeared in Java file

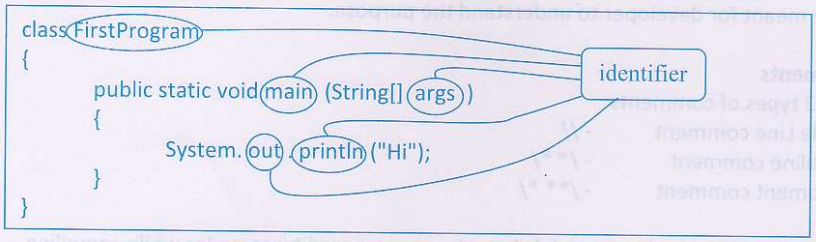
1. //
2. ///////////////
3. // /*
4. /*
5. /* */
6. /* // */
7. // /* // */
8. /* /* */
9. /* */ */
10. // /* */ */
11. /*/* */ */

Identifier and its rules

Definition

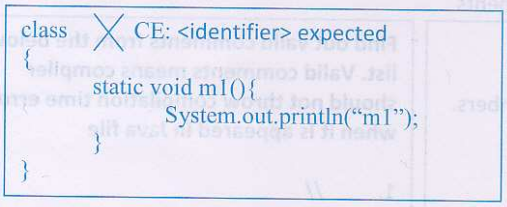
Identifier is a *name* of the basic programming elements.

Find out identifiers in the below program



Compile time error

If we define any basic programming element without name, compiler throws Compile time error: <identifier> expected



Rules in defining identifier

While defining identifier we must follow below rules, else it leads to compile time error.

1. Identifier should only contain
   - i. Alphabets [a to z] and [A to Z]
   - ii. Digits [0 to 9]
   - iii. Special characters [_ or $]

2. Identifier should not start with a digit. A digit can be used from second character onwards
   - Ex: 1stStudent ×
   - No1Student ✓

3. Identifier should not contain special characters, the only '_' or '$' are allowed
   - Ex: No#1Student ×
   - MIN_BALANCE ✓

4. Identifier should not contain space in the middle of words. If we want to provide gap between words, they must be connected with '_'. Due to this reason '_' is called as connector symbol.

Ex: First Program ×
FirstProgram ✓
First$Program ✓
First_Program ✓

5. Identifier is case sensitive (a != A)

```
class A{
   void b(){} ✓
   void b(){} ×
   void B(){} ✓
}
```

6. Keyword cannot be used as user defined identifier, because they are available throughout JVM directly

```
class static{ ×
}
```

Note:

1. Predefined class names can be used for user defined identifier, because they are defined with separate package.
2. There is no limit in identifier length.

Q) When we cannot use keyword as user defined identifier, how can we use predefined class name as user defined identifier?

Keywords are directly available in JVM so if we use them as our user defined identifier we cannot differentiate them from predefined keywords where as predefined classes are available in packages so we can differentiate our classes from predefined class by using package name.

For example: If we create a class with the name String, we must predefined String name with package name "*java.lang.String*".

Run below test case

Create a java file with the name String.java with the class String as shown below

//String.java

```
class String{}
```

Crete a class Test with main method and compile and execute it.

//Test.java

```
class Test{
   public static void main(String[] args){
         System.out.println("Test main");
   }
}
```

Test class is compiled fine but it leads to exception "java.lang.NoSuchMethodError: main", because Java executes main method with the parameter "java.lang.String". In this Test class main method parameter is not java.lang.String, it is our user defined class String.

To execute Test class we must use String class name with package name as "java.lang.String" check Test class in the next page.

//Test.java

```
class Test{
   public static void main(java.lang.String[] args){
         System.out.println("Test main");
   }
}
```

Now this Test class is executed and we will see output *Test main* on console

Now tell me what is the complete prototype of main method?

public static void main(java.lang.String[] args)

Q) In first chapter we have not used package name "java.lang" then how did JVM execute all those classes?

Very simple reason, in that chapter we did not create a class with name String. So compiler automatically places package "java.lang" the String parameter.

Find out valid identifiers from the below list. Valid identifier means - in program when we use below identifiers compiler should not throw error. Just apply above rules to get answer.

1. hihellohru
2. abc1234
3. 4321cba
4. _____$$$$$$$$
5. static
6. firstprogram
7. Class
8. main
9. String

Java Keywords and its rules

Definition

Keywords are predefined identifiers available directly throughout the JVM. They have a special meaning inside Java source code and outside of comments and Strings.

For Example: public, static, void, class etc

Rules

1. Keywords cannot be used as user defined identifier by the programmer either for variable or method or class names, because keywords are reserved for their intended use.
2. All characters in keyword must be used in lower case, because identifier is case sensitive.

Q) Is Class a keyword?

No, it is not a keyword. Its first character is in uppercase.

Need of keywords

Basically keywords are used to communicate with compiler and JVM about the operations we are performing in Java application.

In a Java file, we perform 10 different operations using keywords

They are:

01. Creating Java file (class, interface, or enum)
02. Storing data temporarily with different size of memory locations
03. Creating memory locations
04. Controlling calculations and modifications
05. Setting accessibility permissions
06. Modifying other default properties
07. Establishing relations between classes
08. Object representation
09. Grouping classes
10. Handling user mistakes

In JAVA, we have 50 keywords to perform all above 10 operations.
Among them 47 were introduced in Java 1.0

- In Java 1.2 new keyword "*strictfp*" was added
- In Java 1.4 new keyword "*assert*" was added
- In Java 5 new keyword "*enum*" was added

Among 50 keywords 2 keywords are reserved words they cannot be used in java program, because keyword is defined but they are not implemented, those are "const, goto".
Below is the list of all 50 keywords. I have divided all 50 keywords into 10 categories based on the operation they perform for easily remembering.

1. Java files (3)
   1. class
   2. interface
   3. enum (1.5)

2. Data Types (8 + 1)
   4. byte
   5. short
   6. int
   7. long
   8. flaot
   9. double
   10. char
   11. boolean
   12. void

   These 8 keywords can be used as *data types* and also as *return tye*

   This keyword can only be used as *return type*

3. Memory Location (2)
   static
   14. new

4. Control Statements (11)
   1. conditional
      15. if
      16. else
      17. swith
      18. case
      19. default
   2. loop
      20. while
      21. do
      22. for
   3. transfer
      23. break
      24. continue
      25. return

5. Accessibility Modifiers (3)
   26. private
   27. protected
   28. public

6. Modifiers (8)
   29. static
   30. final
   31. abstract
   32. native
   33. transient
   34. volatile
   35. synchronized
   36. strictfp

7. Inheritance relationship
   40. extends
   41. implements

8. Object representation
   37. this
   38. super
   39. instanceof

9. package
   41. package
   42. import

10. Exception Handling(5 + 1)
    43. try
    44. catch
    45. finally
    46. throw
    47. throws
    48. assert (1.4)

11. Unused keywords
    49. const
    50. goto

These are not keywords

Default literals

1. referenced literal
   -> null
2. boolean literals
   -> true
   -> false

Answer below questions

Q1) Is null a keyword?
A) No, it is not a keyword. It is a literal.

Q2) Then can we use null as user defined identifier?
A) No, even though it is not a keyword we cannot use it as identifier.

Q3) How many datatype keywords Java supports? A) 8

Q4) How many types of datatypes java supports?
A) 2 types
1. Primitives types (8)
2. Referenced types (4)

Q5) How many modifiers java supports? A) 11

Q6) How many Accessibility levels java supports?
A) Four, among them 3 are keywords.
For more details check accessibility modifiers chapter.

Find out valid keywords from the below list

01. static	10. instanceOf
02. final	11. strictf
03. syncronize	12. Void
04. package	13. String
05. imported	14. dowhile
06. Public	15. Enumeration
07. main	16. null
08. object	17. finalize
09. void	18. sizeOf

Q) What is the difference between final, finally and finalize?
final and finally are keywords where as finalize is a method name.

final
is a keyword used to create constant variable, method, and class

finally
is a keyword used to define a block of statements to be executed definitely for a try block.

finalize
is a method used to define logic to be executed definitely before an object is destroyed.

The logic we write inside finally block and finalize method is called resource releasing logic or clean-up code.