

3. Values And Data Types

4. Identifiers

1. Various Identifiers

Identifiers are the **names** given to **packages, classes, methods, variables, objects, arrays** etc. In the examples given below identifiers are in bold letters:

- | | |
|--|---|
| 1) import java.util.* ; | java.util : pre-defined package name |
| 2) public class Sum | Sum : user-defined class name |
| 3) void main() | main() : method name |
| 4) int x ; | x : user-defined variable name |
| 5) Scanner sc = new Scanner(System.in); | Scanner : pre-defined class name
sc : user-defined object name
System.in : pre-defined object name
System : pre-defined class name |
| 6) sc.nextDouble() ; | nextDouble() : pre-defined method name |
| 7) System.out.println() ; | System.out.println() : Full name of a pre-defined method
System.out : Full name of pre-defined object |
| 8) int a [] = new int[5]; | a : Array name |

2. Pick up the identifiers (user-defined and pre-defined) from the following:

```
import java.util.*;
public class Square
{
    public static void main()
    {
        Scanner sc=new Scanner(System.in);
        double s=sc.nextDouble();
        double a=s*s;
        System.out.println("Area: "+a);
    }
}
```

Answer:

- | | |
|--------------------------|----------------------------|
| 1. java.util | pre-defined package name |
| 2. Square | user-defined class name |
| 3. main() | method name |
| 4. Scanner | pre-defined class name |
| 5. sc | user-defined object name |
| 6. System.in | pre-defined object name |
| 7. s | user-defined variable name |
| 8. nextDouble() | pre-defined method name |
| 9. a | user-defined variable name |
| 10. System.out.println() | pre-defined method name |

3. Naming Rules of Identifiers in General

- Identifiers **can have only alphabets, digits, underscore and dollar sign**. (Other characters like **space - , /** etc. are not allowed. **It should be single word**).
Phone Bill (*Invalid*) PhoneBill (*Valid*)
Phone-Bill (*Invalid*) Phone_Bill (*Valid*) Phone\$Bill (*Valid*)
H.C.F. (*Invalid*) HCF (*Valid*)
- Must not begin with digits**.
10B (*Invalid*) B10 (*Valid*) _10B (*Valid*) \$20 (*Valid*)
- Should not be a keyword or reserved word**.
public (*Invalid*) Public (*Valid*)
class (*Invalid*) Class (*Valid*)
- Same variable name should not be declared multiple times**.
int **a**; double **a**; are *invalid* but int **a**; double **A**; are *valid* (identifiers are case sensitive).
- Can be of any length**.
total_sales_amt_of_Jan_2022

Write whether the following identifiers are valid or invalid:

- | | |
|----------------|------------------------------------|
| 1) Income Tax | 6) 4G |
| 2) IncomeTax | 7) _4G |
| 3) Income-Tax | 8) class |
| 4) Income_Tax | 9) Class |
| 5) Income\$Tax | 10) purchaseAmountOfHouseHoldItems |

Answer

- | | |
|------------|------------|
| 1) Invalid | 6) Invalid |
| 2) Valid | 7) Valid |
| 3) Invalid | 8) Invalid |
| 4) Valid | 9) Valid |
| 5) Valid | 10) Valid |

4. Naming Conventions

- Package names: Small letters: E.g.: java.util
- Class names: First letters of each word in capital letters and others in small letters. E.g.: StudentsDetails
- Method names: First word in small letters and First word of other words in capitals. First word verb and next words identifiers. E.g.: findArea()
- Variable names, object names, array names: Small letters. E.g.: sum

5. Variables

Variables are one of the identifiers. **Names** given to the memory cells are **variable names**.

Values that is stored in a memory cell can be varied (changed) (**Vary+able is Variable**).

E.g.: **int a = 5; a=a+3;** The value of **a** is varied (changed) from **5** to **8**.

1. Definition:

Variable represents a **named memory location** which holds a **value** of a **particular data type** that can be **varied during program run**.

Note the words **particular data type**. Before using a variable we have to mention what kind of data is to be stored in it. For example **int a**; Here it is declared that an **integer** is to be stored in **a**. **double b**; it is said that a decimal pointed value is to be stored in **b**.

2. Various Terms

1. Variable Declaration

E.g.: **int a**; For a **declaration statement** it needs data type (here **int**), variable name (here **a**) and ; (semicolon). Now a statement is formed with three elements.

2. Variable Initialization

E.g.1: **int a = 5**; It is an **initialization statement**. Initial means beginning. In the beginning itself a value is stored in the variable. That is why it is known as initialization statement. When the **program is compiled** the value **5** is stored in variable **a**.

E.g.2: **int a = 5+3**; It is also an **initialization statement**. An **expression** is initialized here. When the **program runs** the **5+3** is calculated and **8** is stored into the variable.

E.g.3: **double n=Math.sqrt(25)**; When **program runs** square root of **25** i.e., **5.0** is found and stored in **n**.

3. Value Assignment to Variable

E.g.: **a=8**; Giving value to a variable declared earlier is known as assignment.

4. Value Input into Variable

E.g.: **int a = sc.nextInt()**; Here no value is given to the variable. Value will be stored when the program runs only from the user, (not when compiled).

5. Value Output from Variable

E.g.: **System.out.println(a)**; To print values to the output screen.

Answer the following:

- Write a declaration statement to store a character in a variable named c
- Write an initialization statement to store 6.5 to a variable named b
- Write an initialization statement to store product of 7 and 4.5 into a variable m
- Write an initialization statement to store value of a^b into a variable m
- Write an assignment statement to store 3 to a variable x which is declared already.
- Write an input statement to store a name to a variable na
- Write a statement to display the value in variable m

Answers:

- char c;
- double b=6.5;
- double m=7*4.5;
- double m=Math.pow(a,b);
- x=3;
- String na=sc.next(); or String na=sc.nextLine();
- System.out.println(m); or System.out.print(m);

3. Initialization with Null Values

Null means nothing. It is to be noted that **0** is nothingness. Consider the following:

```
int a;
```

```
System.out.println(a);
```

The above statements causes a **syntax error**. The variable **a** has no value to be printed.

Its correction is:

```
int a=0;
```

```
System.out.println(a);
```

- Every variables must have a value before calculating or printing it.
- A primitive variable has no value when it is declared.
- When it is tried to process or print a variable which has no value in it causes a syntax error.
- So the variable must be initialized with zero or null value explicitly.

Correct the errors (if any):

1. **int a; a++;**
2. **int x; x+=3;**
3. **double m;**

```
System.out.println(m+2.0);
```

Answer:

1. **int a=0; a++;**
1. **int x=0; x+=3;**
2. **double m=0.0;**

```
System.out.println(m+2.0);
```

4. Null Value Initialization to Different Data Types

byte	0	short	0
int	0	long	0L or 0I (small I)
float	0.0F or 0.0f or 0f or 0F		
double	0.0D or 0.0d or 0.0		
char	'\0' or '\u0000'		
boolean	false		
String	null or "" //Note that there is no space in between ""		

5. Default Data Type

If we write simply **2** it is considered as **int**. If it is **2.0** it is **double**.

Fill up the blank with proper data type:

- 1) ---- a = 2; 2) ---- b = 2F; 3) ---- c = 3.0;

Answer:

- 1) **int** a = 2; 2) **float** b = 2F; 3) **double** c = 3.0;

6. Variable Scope

Scope (accessibility/usability) of a variable in various **program regions** is known as **variable scope**. A **program region** is specified by **{ }**. Program regions are class definitions, method definitions, switch block, if else, and loops.

Scope Rule

A variable declared inside a pair of **{ }** (braces) has access (scope) inside that block only. E.g.:

```
int a=sc.nextInt();
int b=sc.nextInt();
if(a>b)
{
    int big=a;
}
else
{
    big=b; //No scope here. Syntax error
}
System.out.println(big); //No scope here. Syntax error
```

The variable **big** is declared inside the pair of braces of **if** statement. It has no scope outside this braces. The **big** inside the **else** block and print statement out side the braces are considered as different variables and these are not declared.

*Correction: (The **big** should be declared before the if statement.)*

```
int a=sc.nextInt();
int b=sc.nextInt();
int big;
if(a>b)
{
    big=a;
}
else
{
    big=b;
}
System.out.println(big);
```

Now the **big** has **scope** inside the braces as well as outside the braces.

Correct the errors if any:

```
double m=sc.nextDouble();
if(m>=40)
{
    String s= "Passed";
}
System.out.println("Result: "+s);
```

Correction:

```
double m=sc.nextDouble();
String s=""; //The "" is because no else for if
if(m>=40)
{
    s= "Passed";
}
System.out.println("Result: "+s);
```

7. Final Variables

If a variable is declared as final, the value initialized to this variable cannot be changed. The **final** is a keyword. Trying to change the value will cause a **syntax error**. E.g.:

```
final int MAX=100;
MAX+=10;
```

The **MAX+=10**; is invalid because the MAX+=10 means MAX = MAX+10.

Write valid or invalid and reason also:

- 1) **final** int a=5;
a++;

Answer:

The **a++** is invalid.

The value of **a** is **final**; its value cannot be changed. The a++ means a=a+1.

- 2) **final** double x=2.5;
System.out.println(x+7.5);

Answer:

Valid.

The value of **x** is not changing, just printing the added value. But if it is (x+=7.5), it is an error.