Values and Data Types: 4

Lesson 4

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 : *A predfined package name*
2) public class **Sum**
   Sum : *user-defined class name*
3) void **main**()
   main : *A method name*
4) int **x**;
   x : A v*ariable name*
5) Scanner **sc** = new Scanner(System.in);
   Scanner : *A predefined class name*
   sc : *A user-defined object name*
   System.in : *A predefined object name*
   System : *A pre-defined class name*
6) sc.**nextDouble**();
   nextDouble : *A predfined method name*
7) System.out.println();
   System.out.println *: Full name of a predefined method*
   System.out : *Full name of pre-defined object name*
   System : *A pre-defined class name*
8) int **a**[ ] = new int[5];
   a : *Array name*

*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

java.util
Square
main
Scanner
sc
System.in
s
nextDouble
a
System.out.println

Naming Rules of Identifiers in General:

1. 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)*

2. **Must not begin with digits**.

   10B *(Invalid)* B10 *(Valid)* _10B *(Valid)* $20 *(Valid)*

3. **Should not be a keyword** or **reserved word.**

   public *(Invalid)* Public *(Valid)*
   class *(Invalid)* Class *(Valid)*

4. **Same variable name should not be declared multiple times.**

   int a; double a; *are **not valid** but* **int a; double A;** *are valid because identifiers are case sensitive.*

5. Can be of **any length**.

   total_sales_amt_of_Jan_2022

*Write following identifiers are **valid** or **invalid**:*

1) Income Tax
2) IncomeTax
3) Income-Tax
4) Income_Tax
5) Income$Tax

Answer

1) Invalid
2) Valid
3) Invalid
4) Valid
5) Valid

Write following identifiers are valid or invalid:

6) L.C.M.
7) LCM
8) 4G
9) _4G
10) $50

Answers

6) Invalid
7) Valid
8) Invalid
9) Valid
10) Valid

Write following identifiers are valid or invalid:

11) class
12) Class
13) case
14) Case
15) sum

Answers

11) Invalid (keyword)
12) Valid
13) Invalid (keyword)
14) Valid
15) Valid

Write following identifiers are valid or invalid:

16) 'x'
17) double
18) Double
19) pamt
20) purchaseAmountOfHouseHoldItems

Answers

16) Invalid (character literal)
17) Invalid (keyword)
18) Valid
19) Valid
20) Valid

Variables

Variables are one of the identifiers. Values are stored in computer memory. Memories are designed as numerous small cells. **Names** given to these memory cells are **variable names**.

Why these memory cells are known as variables? Vary+able is Variable. Values that is stored in a memory cell can be varied (changed). See the following example:

int a = 5; **a=a+3**; The value of **a** is varied, it became **8**. That is why it is known as **variable**.

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 stored in it. For example **int a;** Here it is declared (said) 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**.

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.

*Write a **declaration** statement to store a character in a variable named c.*

Answer: char c;

Variable Initialization

E.g.: **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**.

*Write an **initialization** statement to store **6.5** to a variable named **b**.*

Answer: double b=6.5;

Dynamic Initialization

E.g.1: **int a = 5+3;** It is known as **dynamic initialization statement**. An **expression** is initialized here. The value **8** is not stored when this statement is compiled. When **running the program** the **5+3** is calculated and **8** is stored into the variable.

E.g.2: **double n=Math.sqrt(25);** When running the program square root of **25** is found and **5.0** is stored in **n**.

*Write a **dynamic initialization** statement to store product of 7 and 4.5 into a variable **m***

Answer: double m=7*4.5;

*Write a dynamic initialization statement to store value of a^b into a variable **m***

Answer: **double m=Math.pow(a,b);**

Value Assignment to Variable

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

*Write an assignment statement to store **3** to a variable named **x** which is declared before.*

Answer: x=3;

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).

*Write an input statement to store a **name** to a variable **na***

Answer: String na=sc.next(); or String na=sc.nextLine();

Value Output

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

*Write a statement to display the value in variable **m***

Answer: System.out.println(m);

Practice Questions

Write statements for the following:

1. Declare a variable named **a** that can store **5**
2. Assign **6** to the variable **a**.
3. Initialize a real number to a variable named **b**.
4. Write statement to output the value in **b**.
5. Input a character in variable **c**.

Answers:

1. int a;
2. a=6;
3. double b=3.5;
4. System.out.println(b);
5. char c=sc.next().charAt(0);

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. During program run when it is tried to calculate or to print values, it causes a syntax error as there is no value in it. So it must be initialized with zero or null value explicitly.

Write valid or invalid. If it is invalid correct the errors :

int a;

a++;

Answer:

Invalid.

Correction:

int a=0;

a++; ***//*a=a+1;**

Null Value Intialization to Different Data Types

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

Default data type

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

Correct the errors (if any):

1. int x;
   x+=3;
2. double m;
   System.out.println(m*2.0);

Answer:

1. int x=0;
   x+=3;
2. double m=0.0;
   System.out.println(m*2.0);

Variable Scope

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

It causes **syntax error**. The **big** is declared inside braces. It has no scope outside the braces. The **big** inside the else block and print statement is considered as different one and they are not declared.

Correction:

```
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.

Definition:

Scope (accessibility/availability) of a variable in various **program regions** is known as **variable scope**.

A program region is specified by **{ }**.

A class is a program region:

```
class Sum
{
}
```

A method is a program region:

```
public static void main()
{
}
```

The switch is a program region:

```
switch()
{
}
```

The **if else, loops** etc. also program regions. Even if we do not use curly braces with if else and loops it also will be considered as program regions.

Correct the errors:

```
double m=sc.nextDouble();
if(m>=40)
{
    String result=“Passed”;
}
else
{
    result=“Failed”;
}
System.out.println(result);
```

Correction:

```
double m=sc.nextDouble();
String result;
if(m>=40)
{
    result= “Passed”;
}
else
{
    result=“Failed”;
}
System.out.println(result);
```

Final Variables

A value to a variable initialized with the keyword **final** cannot be changed after the initialization. Trying to change the value will cause a **syntax error**. E.g.:

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

The **MAX+=10;** is invalid.

Write valid or invalid and reason also:

```
final int a=5;
a++;
```

Answer:

Invalid. Because the value of **a** is **final**, its value cannot be changed.