

Data Types

Built-in data types

A data type is a set of values and a set of operations on those values.

<i>type</i>	<i>set of values</i>	<i>examples of values</i>	<i>examples of operations</i>
<code>char</code>	characters	'A' '@'	compare
<code>String</code>	sequences of characters	"Hello World" "CS is fun"	concatenate
<code>int</code>	integers	17 12345	add, subtract, multiply, divide
<code>double</code>	floating-point numbers	3.1415 6.022e23	add, subtract, multiply, divide
<code>boolean</code>	truth values	true false	and, or, not

You can print these data types on the console:

`System.out.println("Smile!!!")` → Smile!!!

`System.out.println(30)` → 30

`System.out.println(10.5)` → 10.5

`System.out.println(true)` → true

You can also store a data type in a **variable**.

What is a variable?



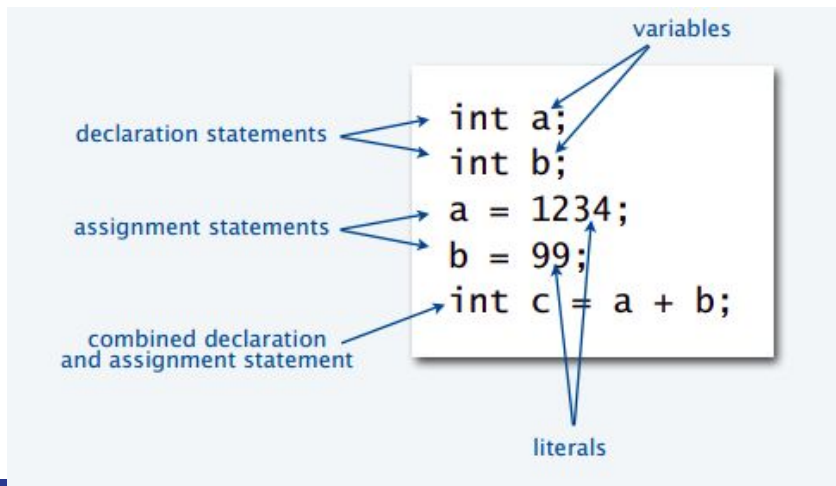
Basic definitions

A **variable** is a name that refers to a value.

A **literal** is a programming-language representation of a value.

A **declaration statement** associates a variable with a type.

An **assignment statement** associates a value with a variable.



Variables always have a name, type and a value.

Declaring a Variable

A variable needs to be created before it is used

`int age;`

Declaration = Data Type + Variable name



Naming Variables

1. Must start with a letter, \$, or _

numGames, \$numGames, _numApples



CORRECT

2Games



WRONG (ERROR)

2. Rest of the name can have letters, numbers or _

numGames2, num_Apples

3. General convention: **lowerCamelCase** (first word in a variable name lowercase, other words should start with uppercase)

numGames

NOTE: Variable name are case sensitive
Variable name is different than variable Name

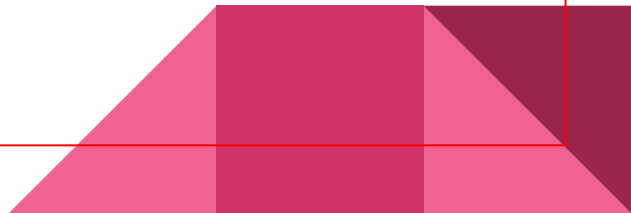
Initializing Variables

A variable can be initialized when it is declared:

```
int numGames = 2;
```

The data type in the declaration must match the assigned value type

```
String numGames = 2; ERROR
```



Initializing Variables

```
int numGames;
```

```
numGames = 2;
```

OR

```
int numGames = 2;
```



Final Variables

Declaring a variable **final** prevents it from being altered.

```
final int numGames = 2;
```

```
numGames = 5; ERROR (variable already assigned)
```

Use final to secure variables especially in complex programs to ensure values do not change.



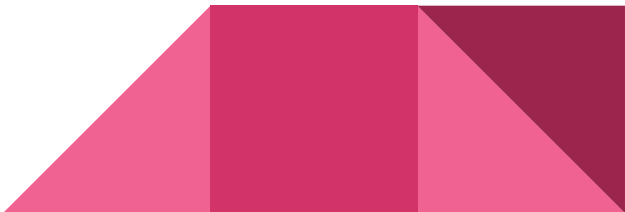
Which of the following is a proper way to declare and initialize a variable in Java?

- a. `myInteger = 100;`
- b. `char = 'a';`
- c. `int myNumber = 10;`
- d. `"Variable"`



Consider the following code snippet:

```
public static void main(String args[]) {  
    final int z;  
  
    z = 20;  
  
    z = 30;  
  
    System.out.println(z);  
}
```

- a. 20
 - b. 30
 - c. This code gives an error.
 - d. Nothing is printed.
- 

What are the memory values associated with the variables **x**, **y**, and **z** after the code snippet below executes?

```
int x = 7;  
  
double y = 2.0;  
  
boolean z = false;  
  
x = x + 3;  
  
z = true;
```

- a. **x** holds the **int** value 7, **y** holds the **double** value 2.0 and **z** holds the **boolean** value false.
- b. **x** holds the **int** value 10, **y** holds the **double** value 2.0 and **z** holds the **boolean** value true.
- c. This code snippet will result in a compile time error.
- d. **x** holds the **int** value 10, **y** holds the **double** value 2.0 and **z** holds the **boolean** value false.

Which of the following variable names follows best practices for naming a variable?

- a. 10movies
- b. numMovies
- c. bestmovies
- d. MyVariable



What does the keyword **final** do?

- a. It's necessary to declare a variable.
- b. Enables the use of `println` on a variable.
- c. It prevents variables from being altered.
- d. It indicates that the program has finished executing.



What will be the output:

```
public class Variables
{
    public static void main(String[] args)
    {
        int totalBirds = 150;

        String mostCommon = "Mallard Duck";


        System.out.println("Bird Watching Results");

        System.out.print("Total birds seen: ");

        System.out.println(totalBirds);


        System.out.print("Most common bird seen was ");

        System.out.println(mostCommon);

    }
}
```

- a. Bird Watching Results
Total birds seen:
150
Most common bird seen was
Mallard Duck
- b. Bird Watching Results
Total birds seen:
150

Most common bird seen was
Mallard Duck
- c. Bird Watching Results
Total birds seen: 150
Most common bird seen was Mallard Duck
- d. Bird Watching Results
Total birds seen: totalBirds
Most common bird seen was mostCommon

Example

```
public class Exchange {  
    public static void main(String[] args) {  
        int a = 1234;  
        int b = 99;  
        int t = a;  
        a = b;  
        b = t;  
    }  
}
```

This code *exchanges*
the values of a and b.

Trace

A trace is a table of variable values after each statement

```
public class Exchange {  
    public static void main(String[] args) {  
        int a = 1234;  
        int b = 99;  
        int t = a;  
        a = b;  
        b = t;  
    }  
}
```

This code *exchanges*
the values of a and b.

	a	b	t
	<i>undeclared</i>	<i>undeclared</i>	<i>undeclared</i>
int a = 1234;	1234	<i>undeclared</i>	<i>undeclared</i>
int b = 99;	1234	99	<i>undeclared</i>
int t = a;	1234	99	1234
a = b;	99	99	1234
b = t;	99	1234	1234

Let's remember the common data types

<i>type</i>	<i>set of values</i>	<i>examples of values</i>	<i>examples of operations</i>
char	characters	'A' '@'	compare
String	sequences of characters	"Hello World" "CS is fun"	concatenate
int	integers	17 12345	add, subtract, multiply, divide
double	floating-point numbers	3.1415 6.022e23	add, subtract, multiply, divide
boolean	truth values	true false	and, or, not

char Type

Represents a single character:

```
char grade = 'A';
```

```
char lastLetter = 'Z';
```

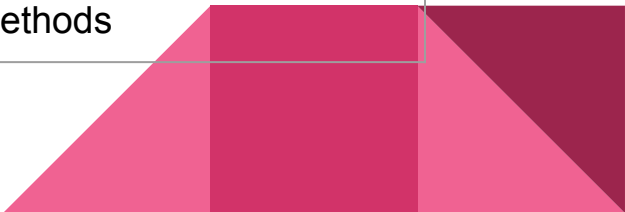
NOTE: char variables must use single quotation and store only one character



Difference between String and a char

Char is a **primitive type**. String is a **reference type**.

Primitive Type	Reference Type
Most basic data type in Java	Instantiable classes made by programmers that often use primitive types
Primitive variables store primitive values	Reference variables store the address of the value
Do not have associated methods	Have associated methods



Storing a Variable

When a primitive value is stored in a variable, it is using memory.

```
char lastChar = 'z';
```

The computer stores this:

lastChar =

0	1	0	1	1	0	1	1
---	---	---	---	---	---	---	---




Primitives vs. References

- Primitive types store the actual primitive values

<code>char firstChar = 'a';</code>	<div><code>firstChar = lastChar;</code></div>	<code>firstChar = 'z'</code>
<code>char lastChar = 'z';</code>		<code>lastChar = 'z'</code>

- Reference types reference an address of the existing value

<code>String firstStr = "hello";</code>	<div><code>firstStr = lastStr;</code></div>	<code>firstStr</code>	<code>"hello"</code>
<code>String lastStr = "friends";</code>		<code>lastStr</code>	<code>"friends"</code>



Which of the choices below is not a primitive type in Java?

1. int
2. char
3. boolean
4. double
5. String

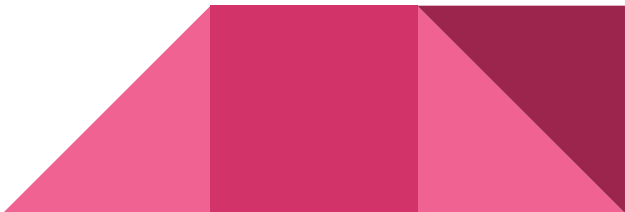


Which of the following is true about primitive types in Java?

1. A variable is the name given to a memory location.
2. The value stored in a variable can be changed during program execution.
3. All variables must be declared before use.
4. All of these statements are correct for primitive variables in Java.



Which of the following could be stored in the variable
`char initial;`

1. `"k"`
 2. `'karel'`
 3. `"karel"`
 4. `'k'`
- 

What is the difference between the `int` type and the `double` type?

1. `int` can be assigned numbers like 1, 3, 3.5, -4, but `double` can only assign numbers like 1, 4, -7, 10.
 2. `int` can be assigned numbers like 1, 3, -4, but `double` can only assign numbers like 1.5, 2.25, -16.987.
 3. `double` can be assigned numbers like 1, 3, 3.5, -4, but `int` can only be assigned numbers like 1, 4, -7, 10.
 4. `double` can be assigned numbers like 1, 3, -4, but `int` can only assign numbers like 1.5, 2.25, -16.987.
- 