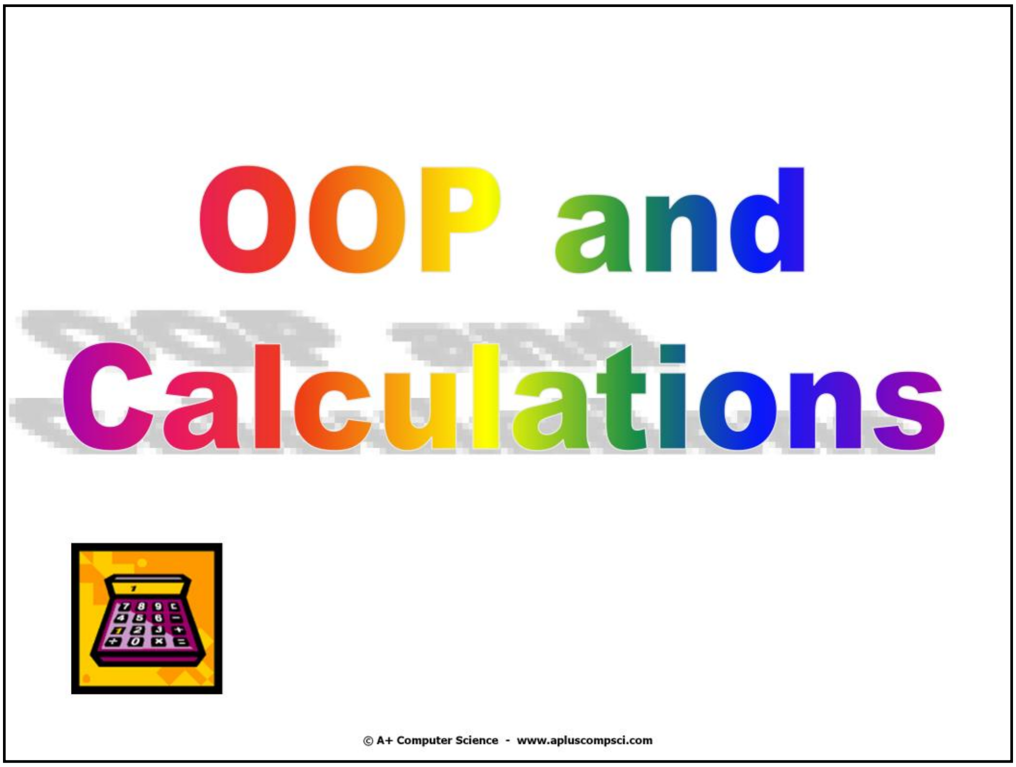
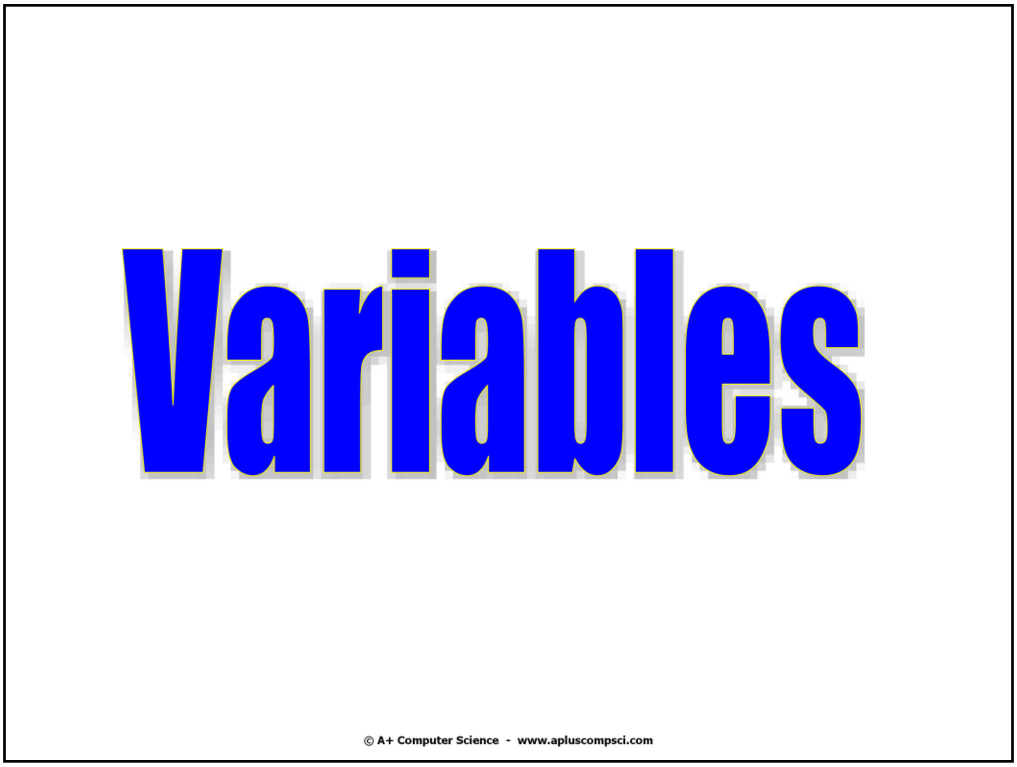
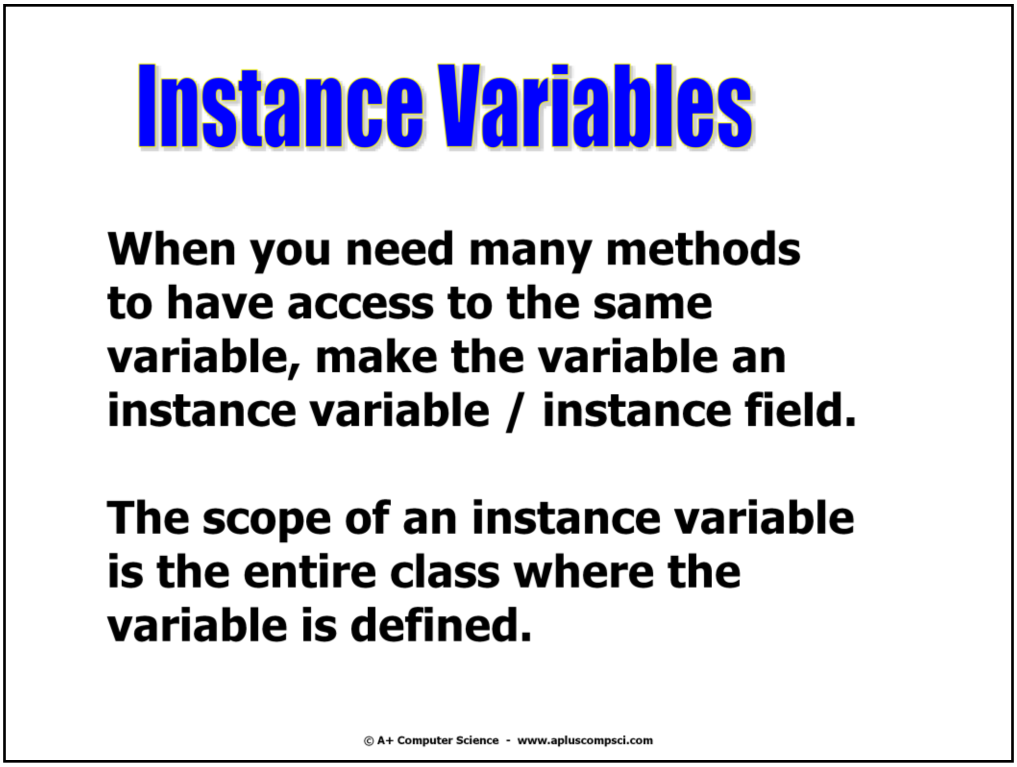


OOP and Calculations



Variables

© A+ Computer Science - www.apluscompsci.com



Instance Variables

When you need many methods to have access to the same variable, make the variable an instance variable / instance field.

The scope of an instance variable is the entire class where the variable is defined.

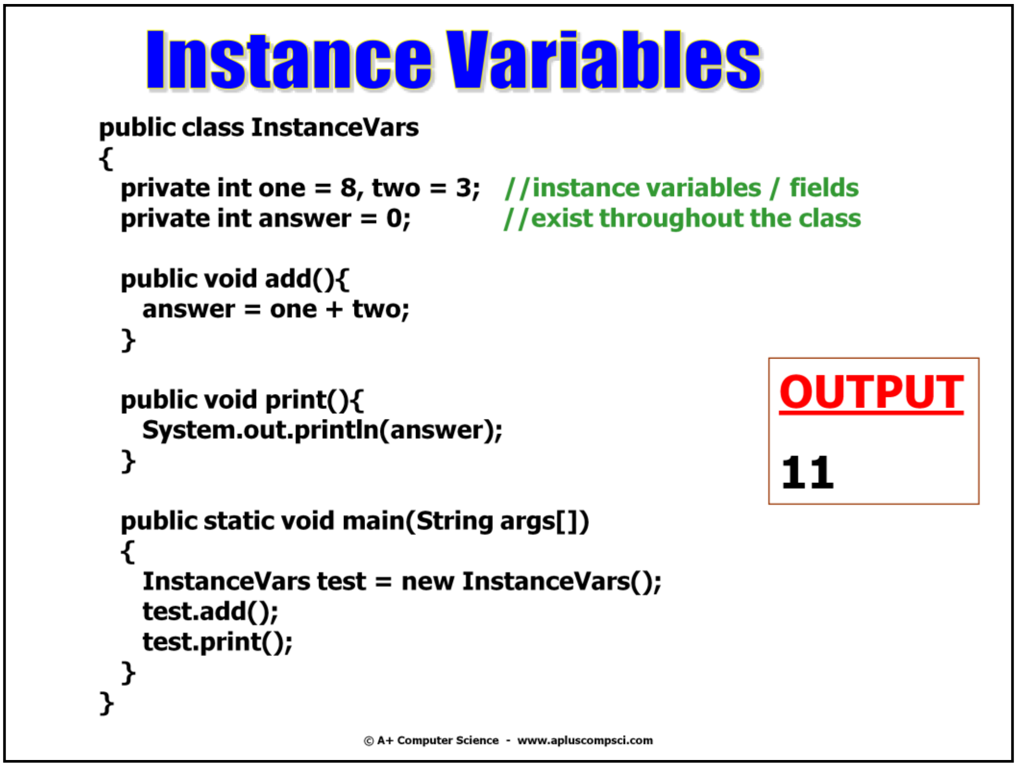
© A+ Computer Science - www.apluscompsci.com

An instance variable is a variable tied to an instance of a class. Each time an Object is instantiated, it is given its own set of instance variables.

Instance variables are also commonly called instance fields.

```
Monster x = new Monster();
```

x refers to a unique Monster that contains its own set of Monster instance variables.



Instance Variables

```
public class InstanceVars
{
  private int one = 8, two = 3;   //instance variables / fields
  private int answer = 0;         //exist throughout the class

  public void add(){
    answer = one + two;
  }

  public void print(){
    System.out.println(answer);
  }

  public static void main(String args[])
  {
    InstanceVars test = new InstanceVars();
    test.add();
    test.print();
  }
}
```

OUTPUT

```
11
```

© A+ Computer Science - www.apluscompsci.com

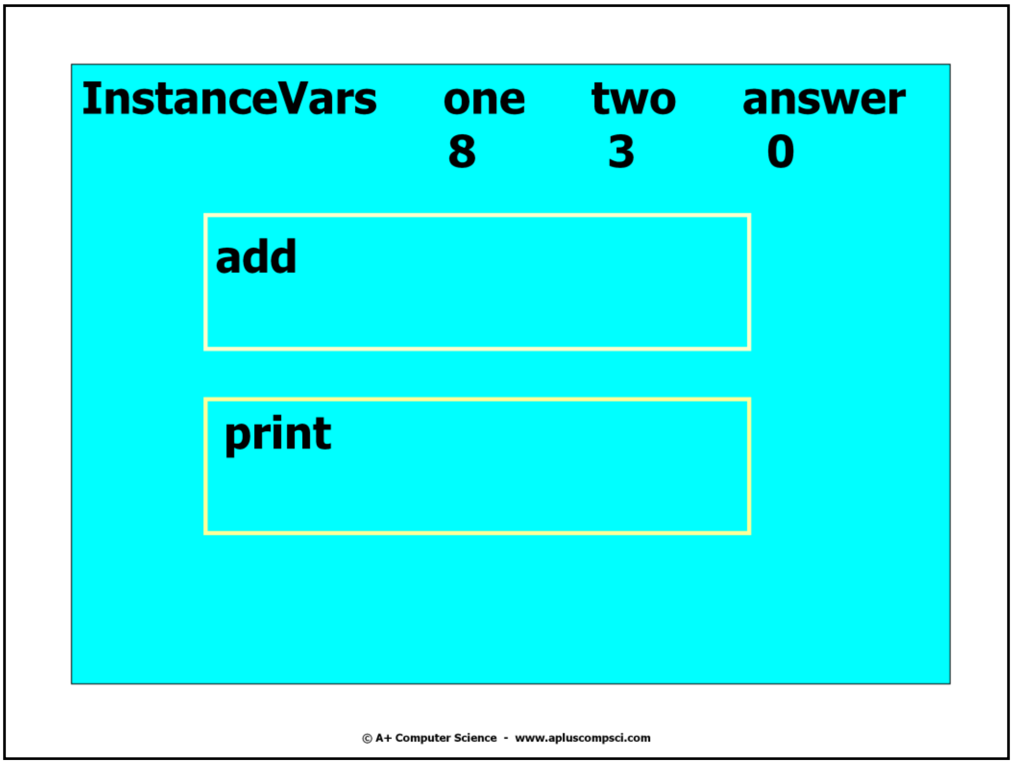
Class `InstanceVars` contains three instance variables : one, two, and total. Each time class `InstanceVars` is instantiated, a new set of instance variables is created inside of the new Object.

```
InstanceVars test = new InstanceVars();
```

test refers to an `InstanceVars` Object that contains one, two, and three.

```
InstanceVars diff = new InstanceVars();
```

diff refers to an `InstanceVars` Object that contains one, two, and three.



InstanceVars one two answer

8 3 0

add

print

© A+ Computer Science - www.apluscompsci.com

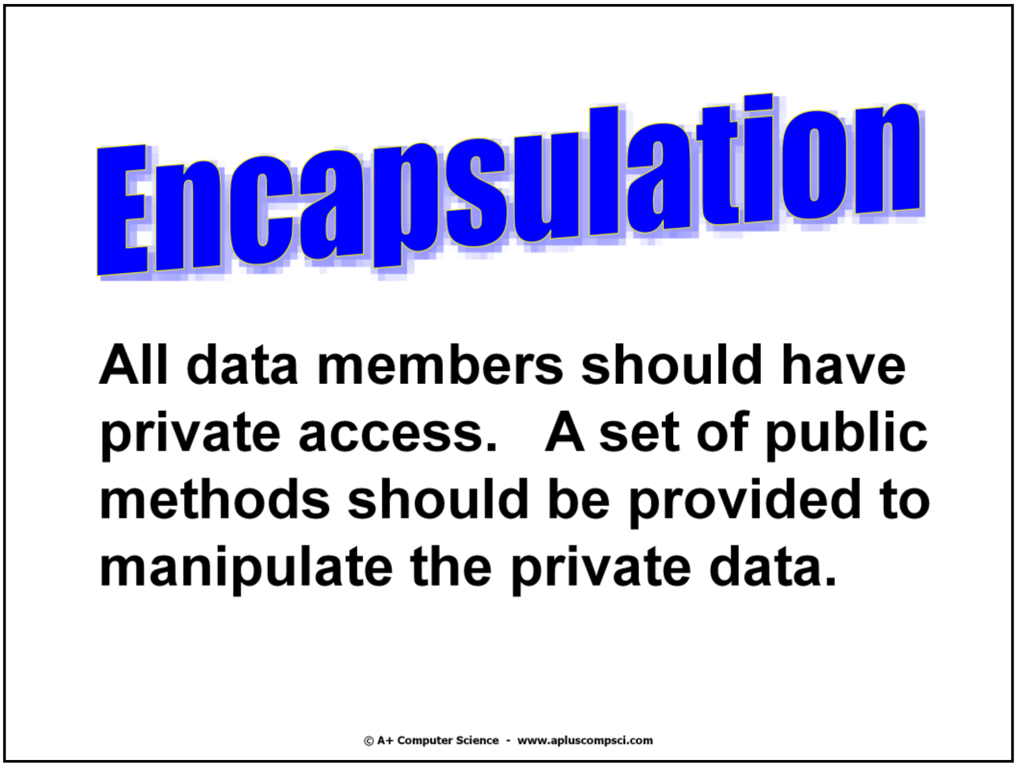
All methods in class `InstanceVars` can access instance vars one, two, and answer.

What does private mean?

All members with private access can be accessed or modified only inside the class where they are defined.

© A+ Computer Science - www.apluscompsci.com

All members of a class with private access can be accessed or modified within the class where they are defined only. Private members cannot be accessed outside of the class.



Encapsulation

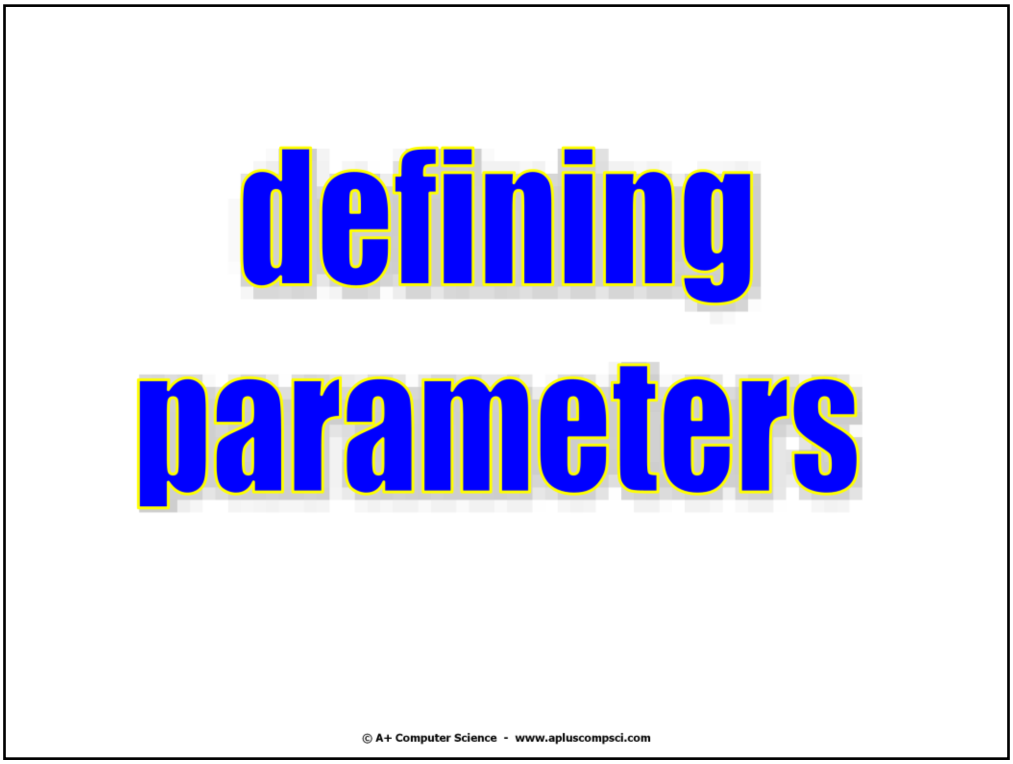
All data members should have private access. A set of public methods should be provided to manipulate the private data.

© A+ Computer Science - www.apluscompsci.com

Data should be declared with private access and public methods should be provided to manipulate the private data.

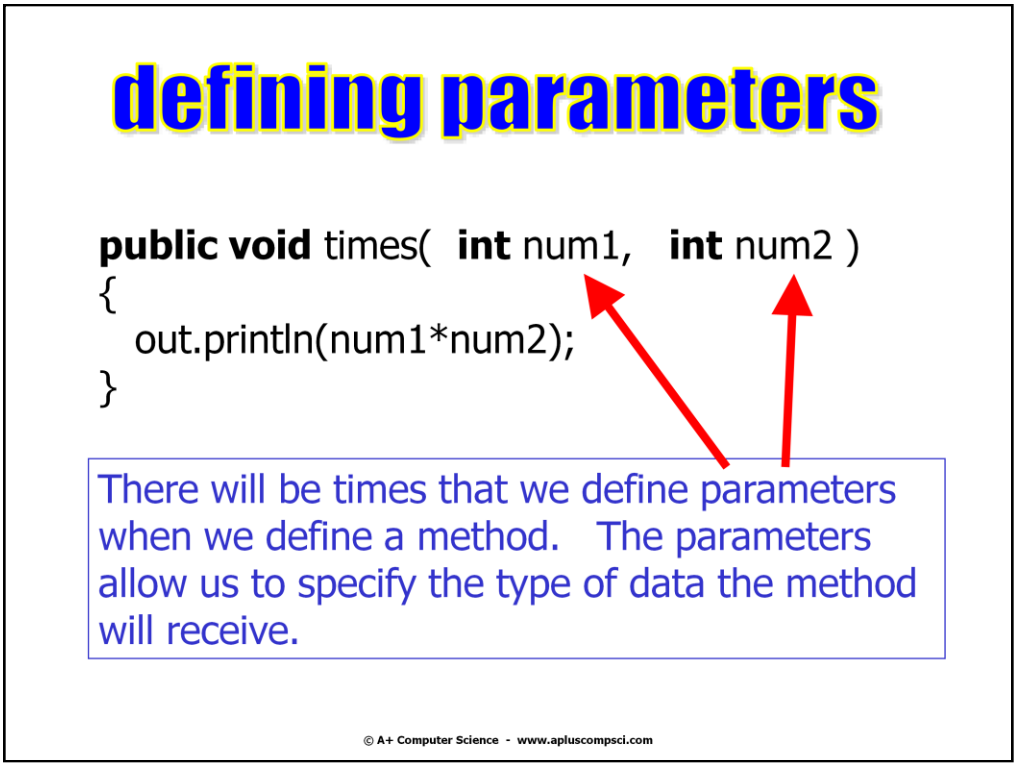
open instancevars.java

© A+ Computer Science - www.apluscompsci.com



defining parameters

© A+ Computer Science - www.apluscompsci.com



defining parameters

```
public void times(  int num1,   int num2 )
{
  out.println(num1*num2);
}
```

There will be times that we define parameters when we define a method. The parameters allow us to specify the type of data the method will receive.

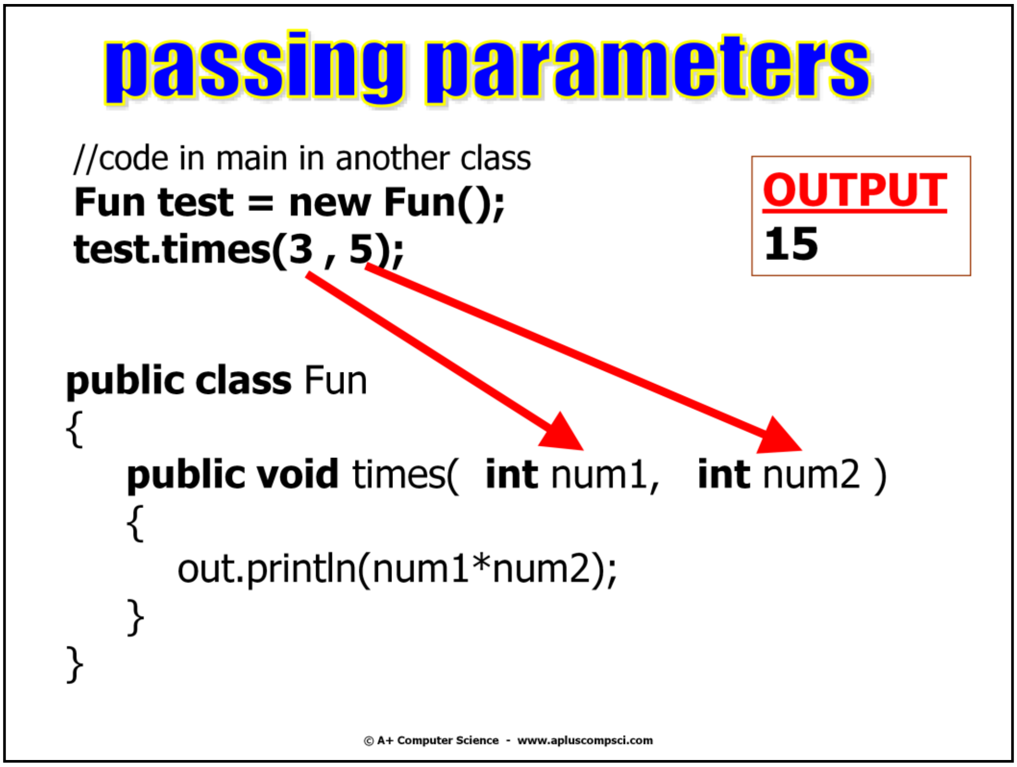
© A+ Computer Science - www.apluscompsci.com

Methods are often defined with a parameter list. Parameters are defined within the parenthesis following the method name.

```
public void method( parameter list )
```

When defining parameters, a data type and name must be provided for each parameter.

```
public void method(int one, double two)

public void go(String word, int num)
```



passing parameters

```
//code in main in another class
Fun test = new Fun();
test.times(3 , 5);


public class Fun
{
     public void times(  int num1,   int num2 )
     {
          out.println(num1*num2);
     }
}
```

OUTPUT

```
15
```

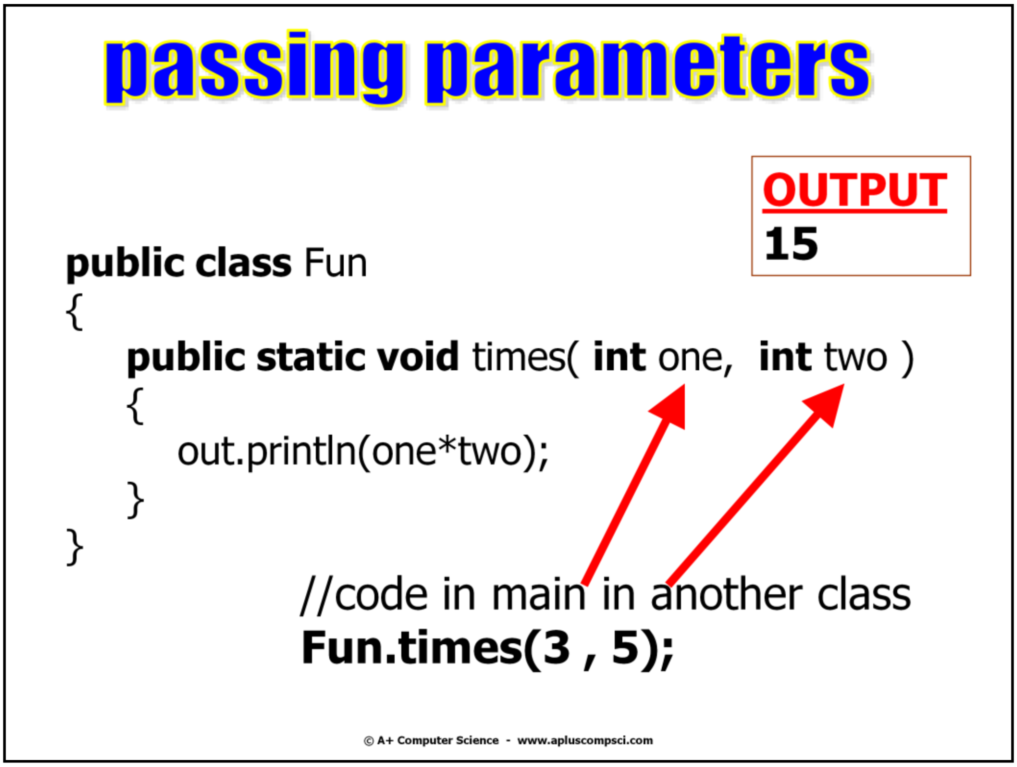
© A+ Computer Science - www.apluscompsci.com

When calling a method with parameters, the data types and number of parameters are very important.

```
public void method(int one, double two)
```

A call to method would have to have 2 parameters.   A call to method would require passing in an integer and a double in that order.

```
method(6, 9.3);
method(562, 32186.323);
```



passing parameters

OUTPUT
15

```
public class Fun
{
   public static void times( int one,  int two )
   {
      out.println(one*two);
   }
}
            //code in main in another class
            Fun.times(3 , 5);
```

© A+ Computer Science - www.apluscompsci.com

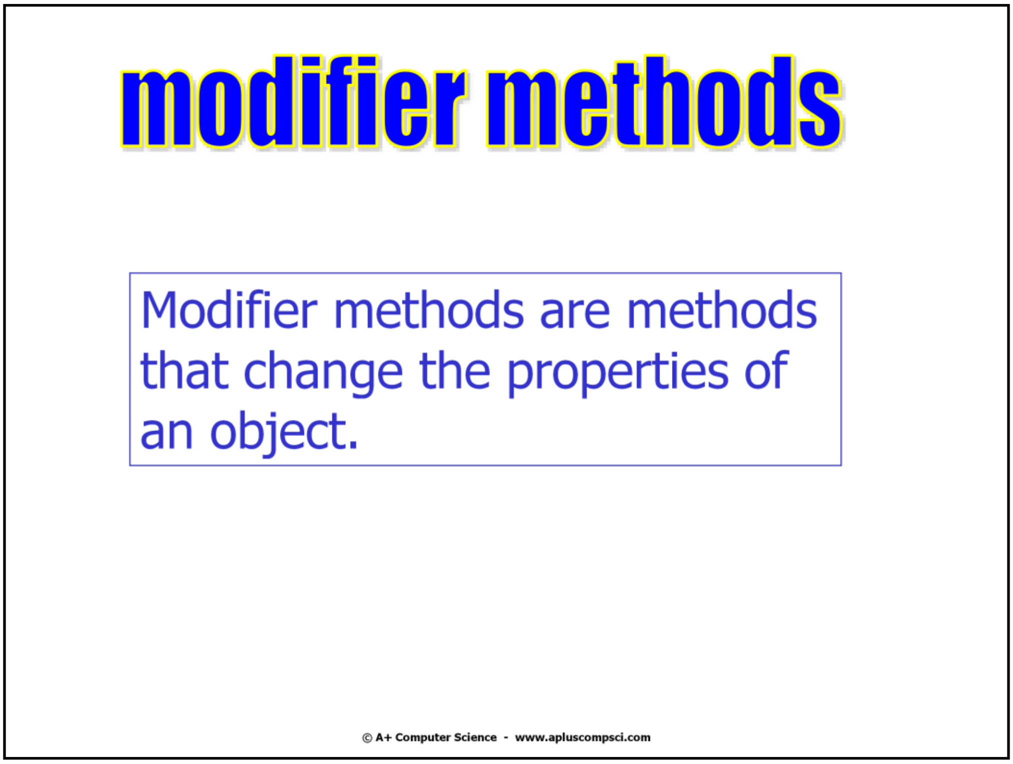
When calling a method with parameters, the data types and number of parameters are very important.

```
public static void times(int one, double two)
```

times is defined as static. A static method exists without the need for an instantiation of the class. Notice that there is no `Fun x = new Fun()` line in the main example.

```
Fun.times(6, 9);
Fun.times(11,22);
```

Open
parametersone.java
parameterstwo.java

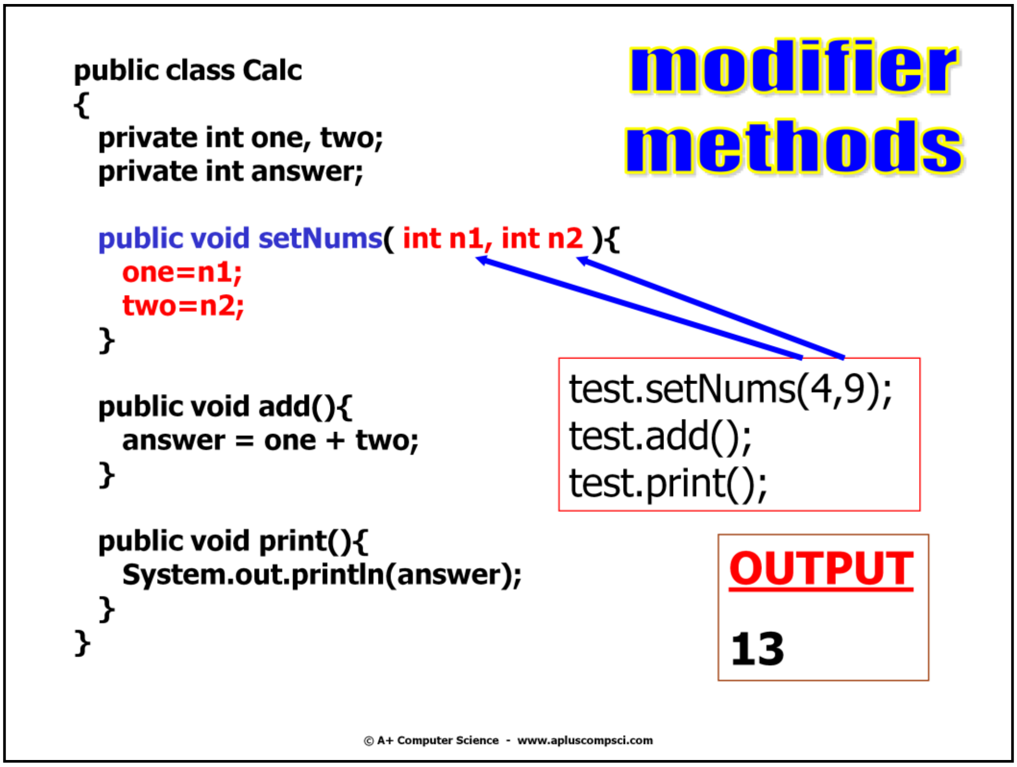


modifier methods

Modifier methods are methods that change the properties of an object.

© A+ Computer Science - www.apluscompsci.com

Modifier methods make changes to the instance variables of the class.



modifier methods

```
public class Calc
{
  private int one, two;
  private int answer;

  public void setNums( int n1, int n2 ){
    one=n1;
    two=n2;
  }

  public void add(){
    answer = one + two;
  }

  public void print(){
    System.out.println(answer);
  }
}
```

```
test.setNums(4,9);
test.add();
test.print();
```

OUTPUT

```
13
```

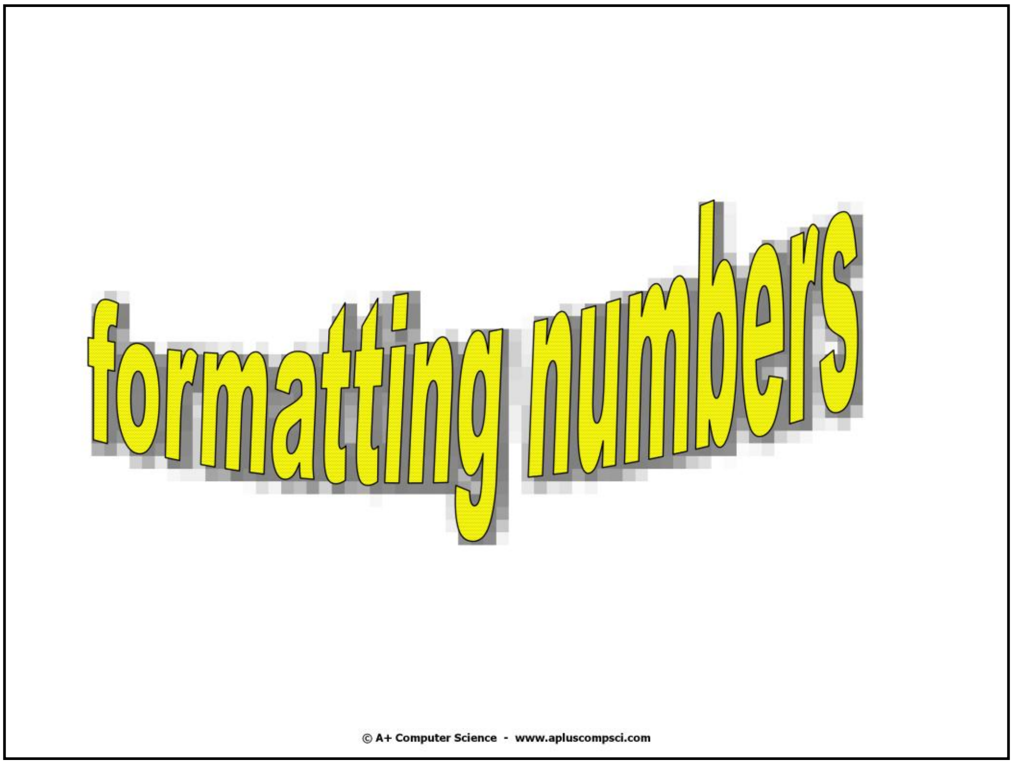
© A+ Computer Science - www.apluscompsci.com

Modifier methods make changes to the instance variables of the class.

Method `setNums()` assigns parameter n1 to instance variable one and parameter n2 to instance variable two.

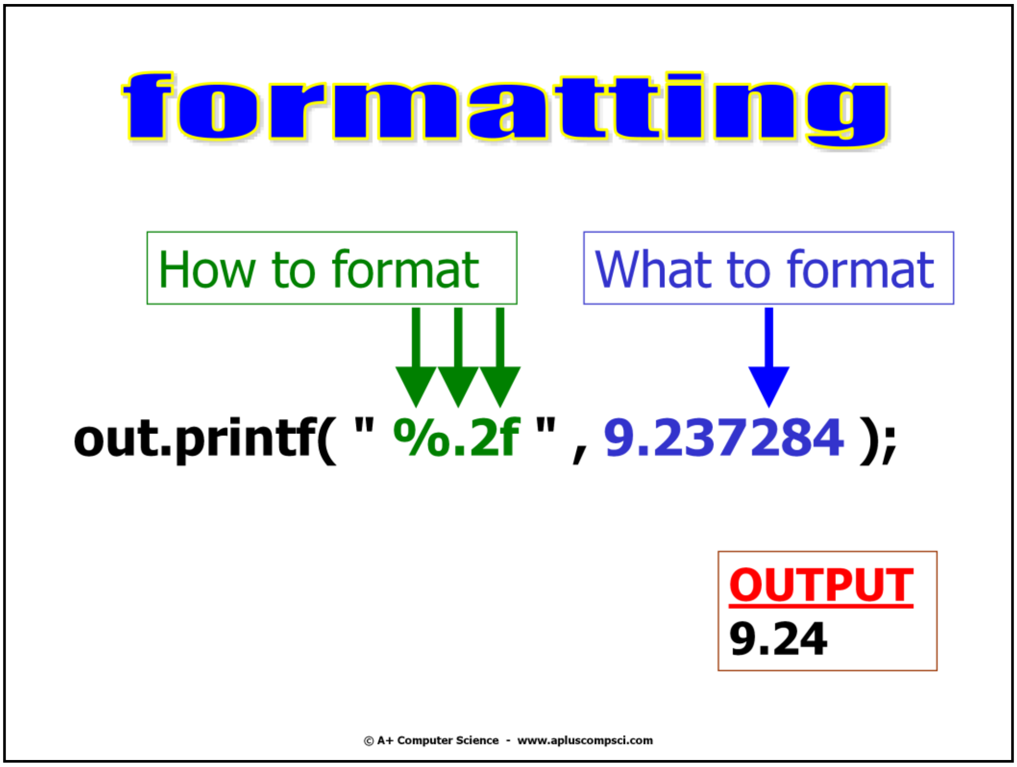
The purpose of method `setNums()` is to modify the instance variables / instance fields.

Open
calc.java
calcrunner.java



formatting numbers

© A+ Computer Science - www.apluscompsci.com



formatting

How to format

What to format

```
out.printf( " %.2f " , 9.237284 );
```

OUTPUT

```
9.24
```

© A+ Computer Science - www.apluscompsci.com

Method `printf()` is used to format output. `printf()` is most commonly used to set the number of decimal places when displaying a real number. `printf()` can also be used to align output to the left or to the right.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

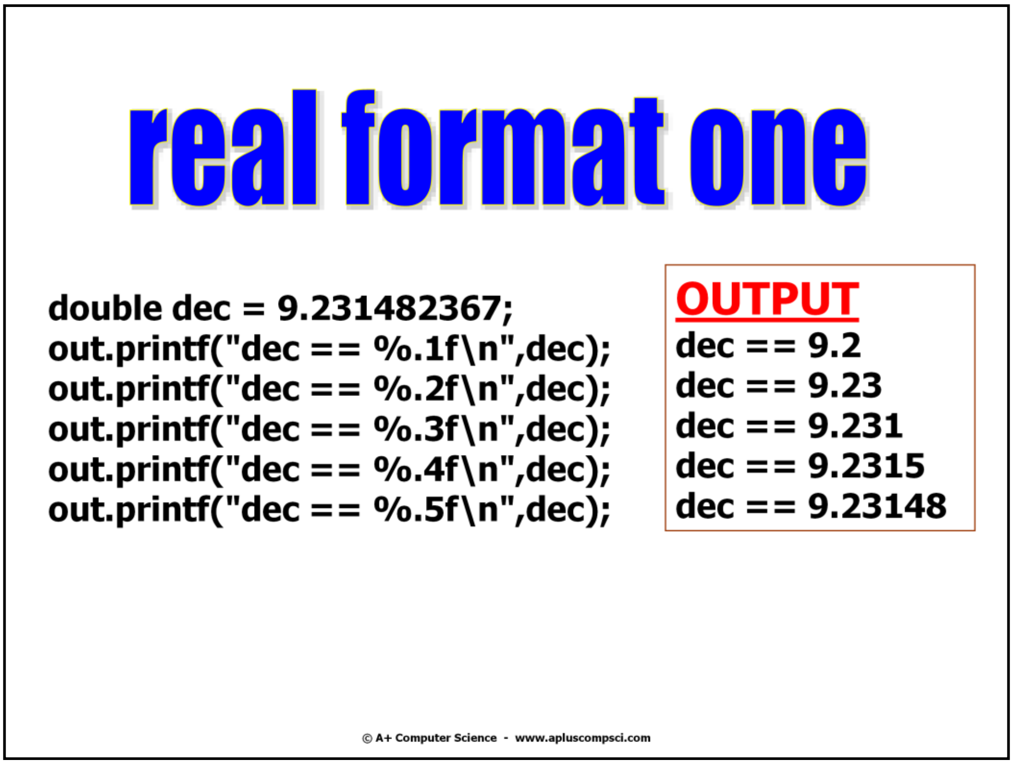
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

`%s` – string value

`-` left aligned



real format one

```
double dec = 9.231482367;
out.printf("dec == %.1f\n",dec);
out.printf("dec == %.2f\n",dec);
out.printf("dec == %.3f\n",dec);
out.printf("dec == %.4f\n",dec);
out.printf("dec == %.5f\n",dec);
```

OUTPUT

```
dec == 9.2
dec == 9.23
dec == 9.231
dec == 9.2315
dec == 9.23148
```

© A+ Computer Science - www.apluscompsci.com

Method `printf()` is used to format output. `printf()` is most commonly used to set the number of decimal places when displaying a real number. `printf()` can also be used to align output to the left or to the right.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

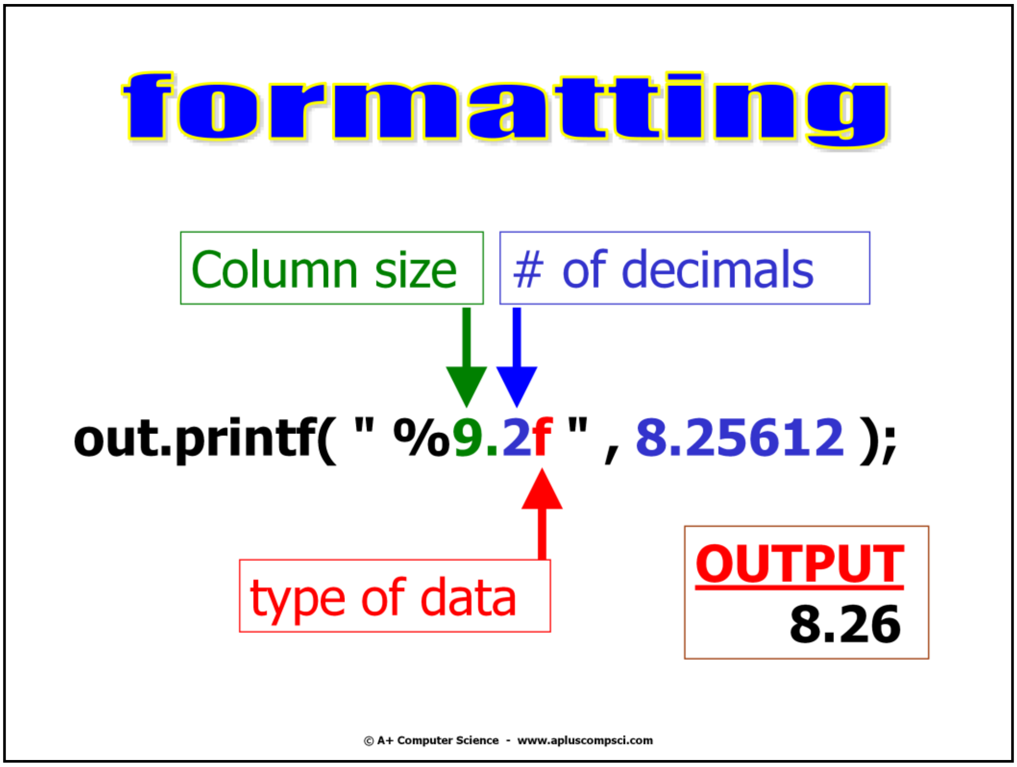
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

`%s` – string value

`–` left aligned



formatting

Column size

# of decimals

```
out.printf( " %9.2f " , 8.25612 );
```

type of data

OUTPUT
8.26

© A+ Computer Science - www.apluscompsci.com

Method `format()` is used to format output. `format()` is most commonly used to set the number of decimal places when displaying a real number.

`format()` differs from `printf()` in that `format()` is a return method and `printf()` is a void method.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

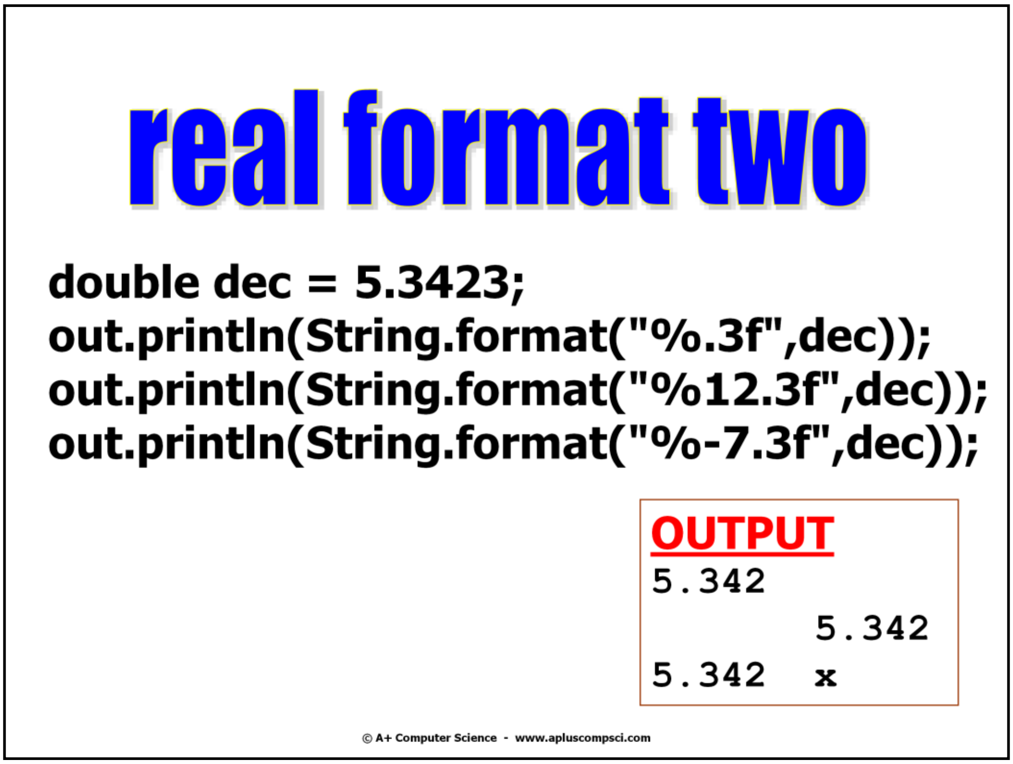
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

`%s` – string value

`-` left aligned



real format two

```
double dec = 5.3423;
out.println(String.format("%.3f",dec));
out.println(String.format("%12.3f",dec));
out.println(String.format("%-7.3f",dec));
```

OUTPUT

```
5.342
       5.342
5.342  x
```

© A+ Computer Science - www.apluscompsci.com

Method `format()` is used to format output. `format()` is most commonly used to set the number of decimal places when displaying a real number.

`format()` differs from `printf()` in that `format()` is a return method and `printf()` is a void method.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

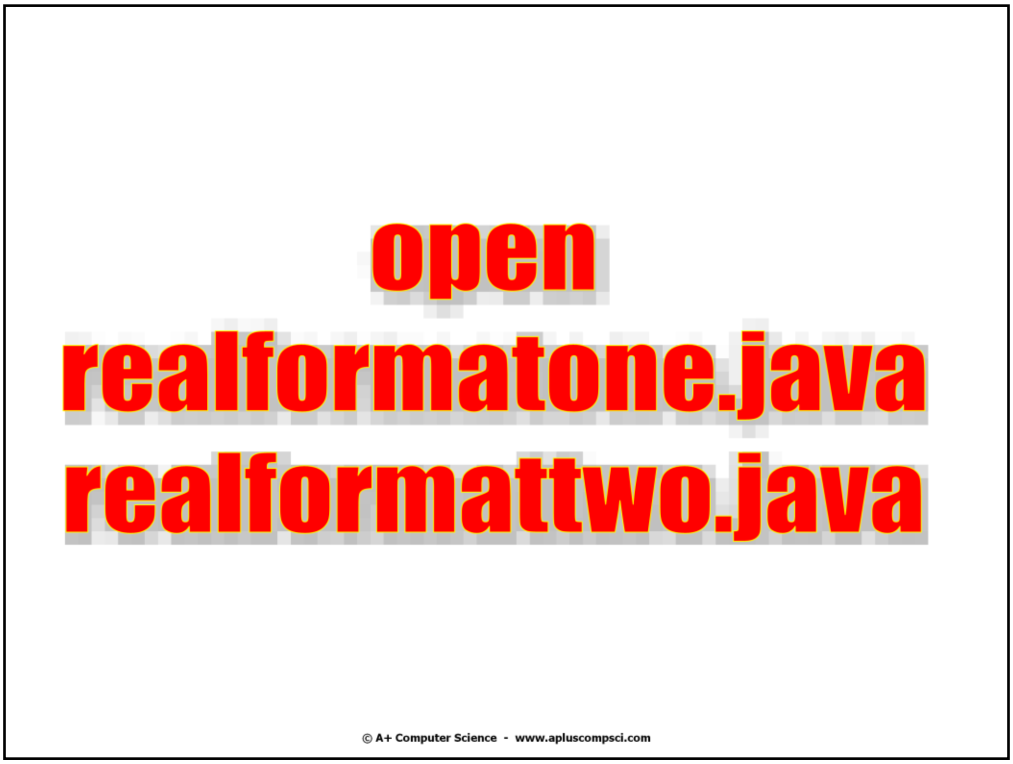
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

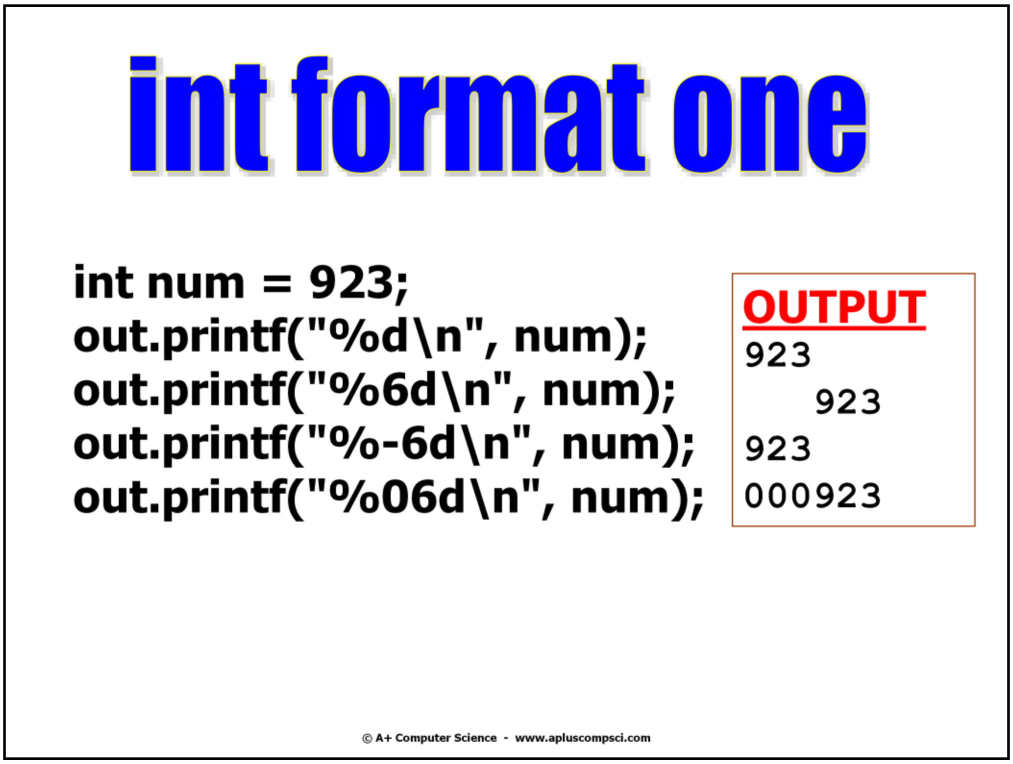
`%s` – string value

`-` left aligned



open
realformatone.java
realformattwo.java

© A+ Computer Science - www.apluscompsci.com



int format one

```
int num = 923;
out.printf("%d\n", num);
out.printf("%6d\n", num);
out.printf("%-6d\n", num);
out.printf("%06d\n", num);
```

OUTPUT

```
923
   923
923
000923
```

© A+ Computer Science - www.apluscompsci.com

Method `printf()` is used to format output. `printf()` is most commonly used to set the number of decimal places when displaying a real number. `printf()` can also be used to align output to the left or to the right.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

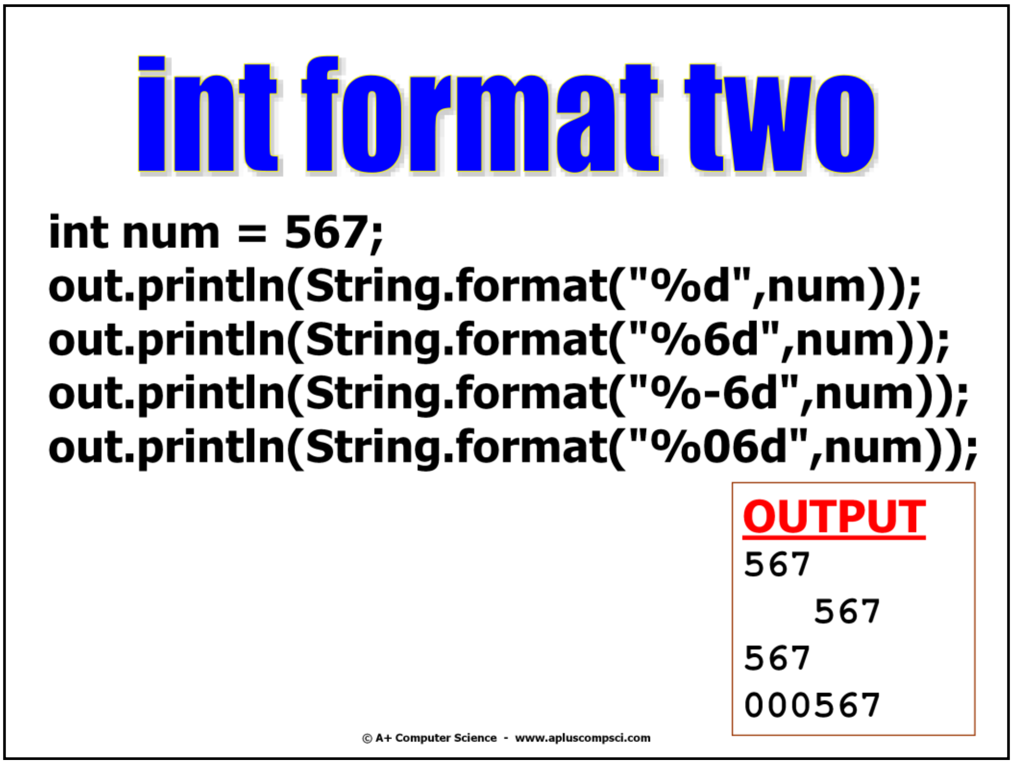
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

`%s` – string value

– left aligned



int format two

```
int num = 567;
out.println(String.format("%d",num));
out.println(String.format("%6d",num));
out.println(String.format("%-6d",num));
out.println(String.format("%06d",num));
```

OUTPUT

```
567
   567
567
000567
```

© A+ Computer Science - www.apluscompsci.com

Method `format()` is used to format output. `format()` is most commonly used to set the number of decimal places when displaying a real number.

`format()` differs from `printf()` in that `format()` is a return method and `printf()` is a void method.

The `%` sign is used to indicate that a value needs to be displayed. The value will be found in the comma separated list.

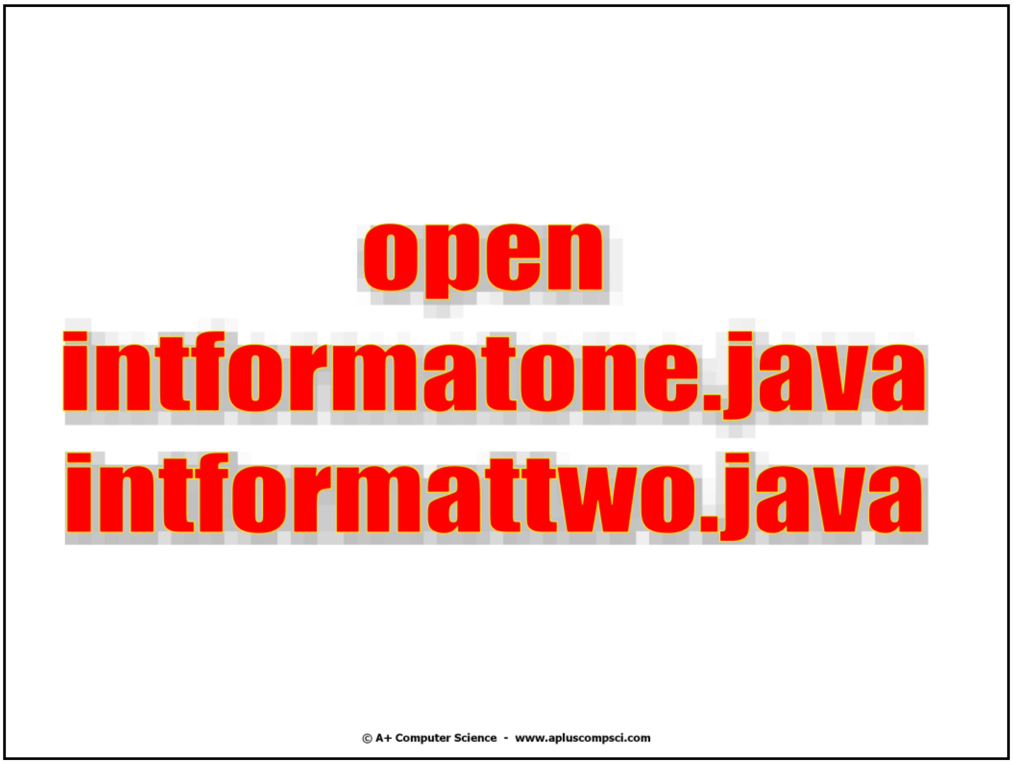
`%f` – real / decimal value

`%d` – integer value

`%c` – character value

`%s` – string value

– left aligned

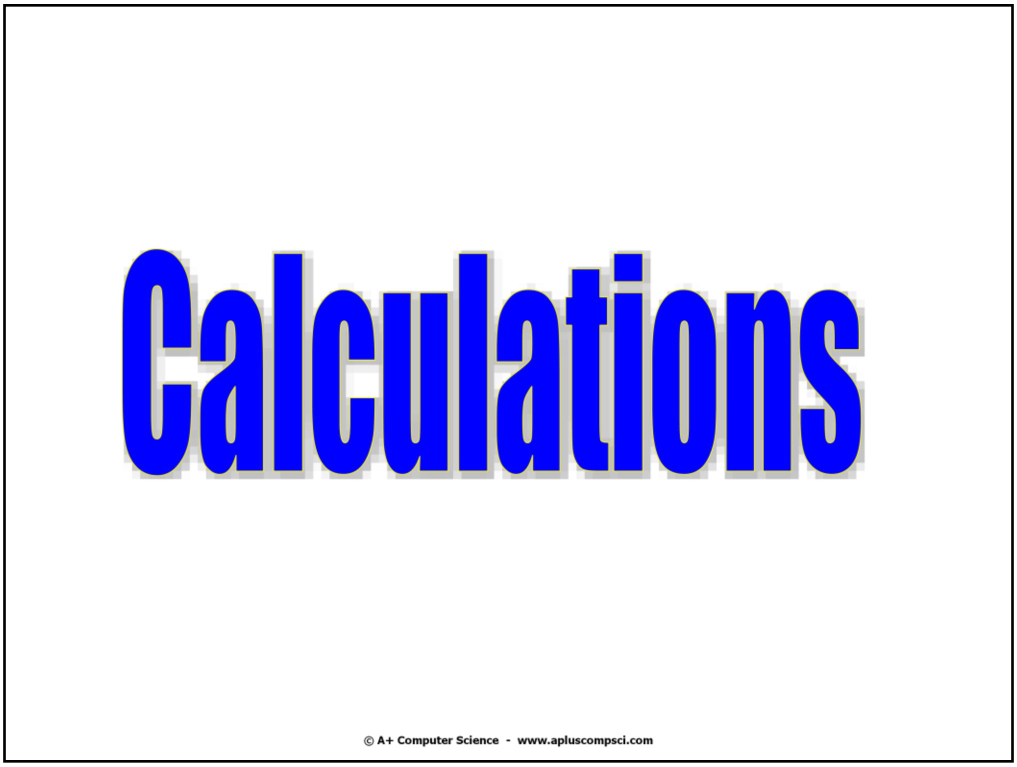


open
intformatone.java
intformattwo.java

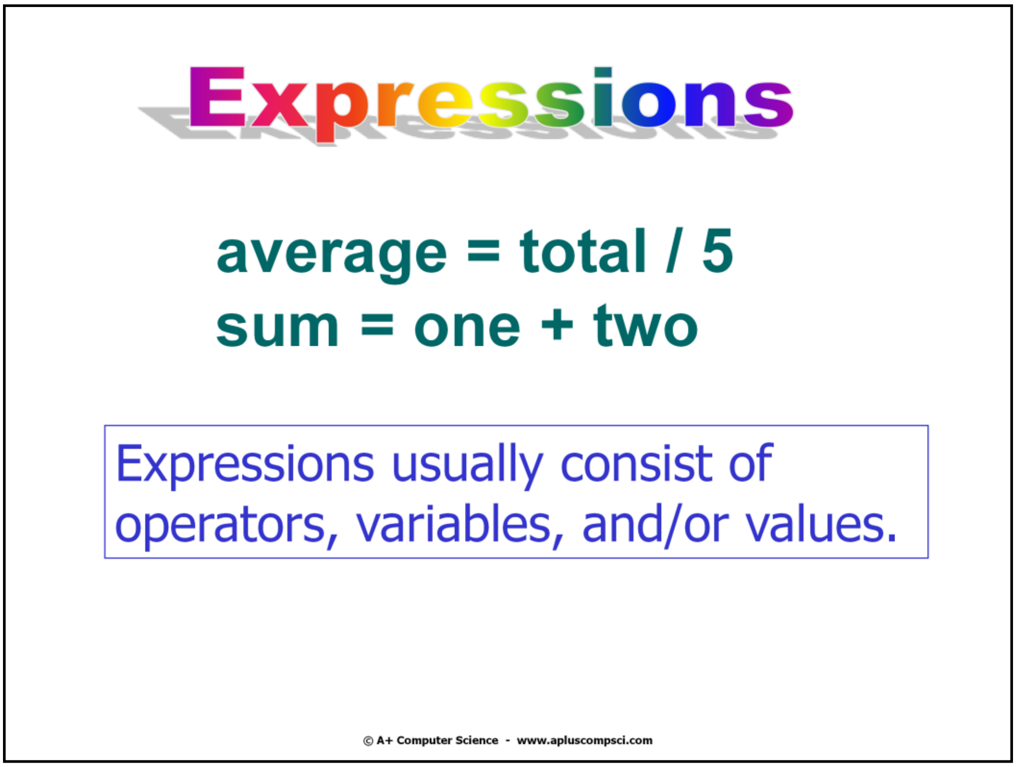
© A+ Computer Science - www.apluscompsci.com

Start work on the labs

© A+ Computer Science - www.apluscompsci.com



Calculations

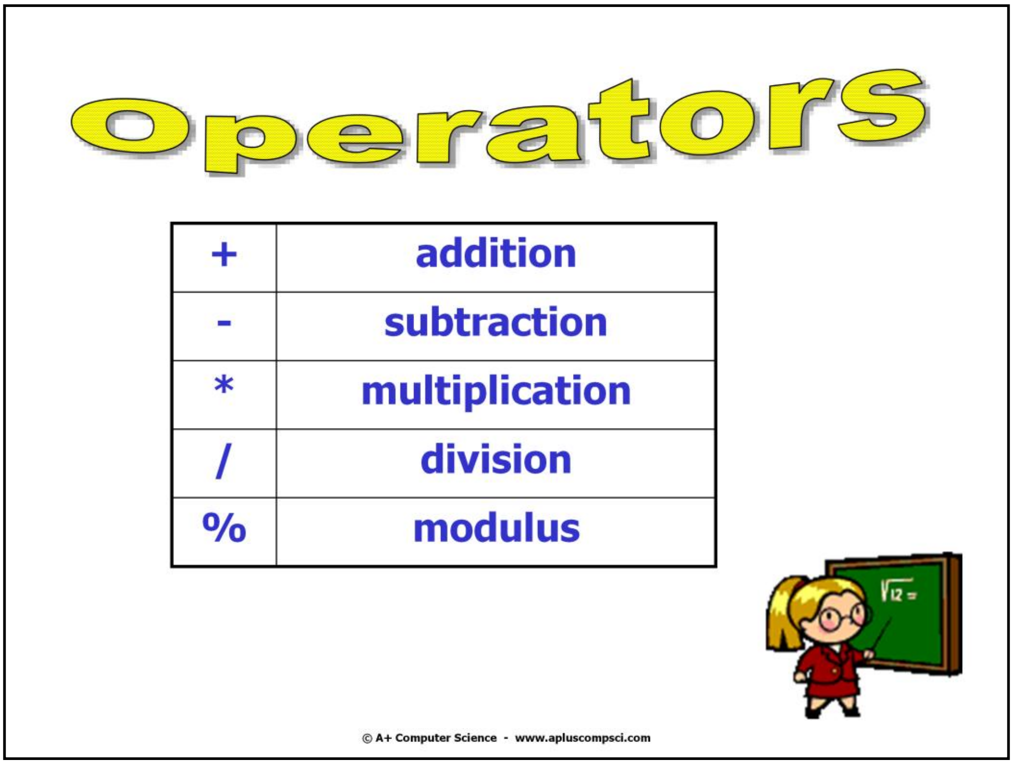


Expressions

```
average = total / 5
sum = one + two
```

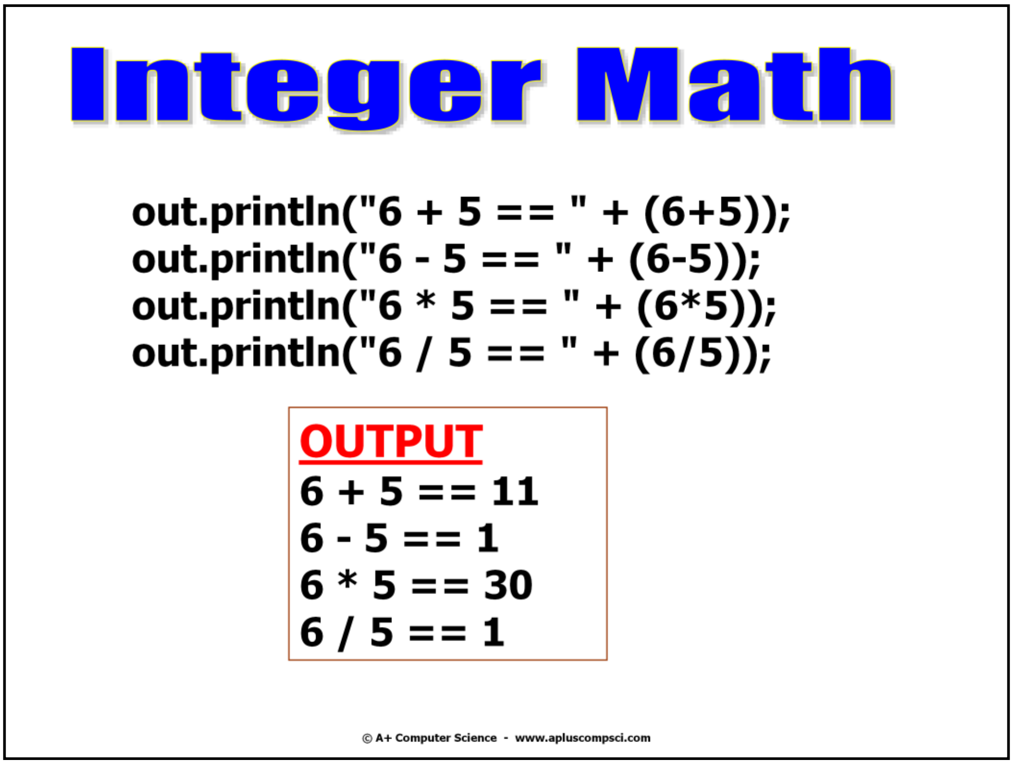
Expressions usually consist of operators, variables, and/or values.

© A+ Computer Science - www.apluscompsci.com



Operators

+	addition
-	subtraction
*	multiplication
/	division
%	modulus



Integer Math

```
out.println("6 + 5 == " + (6+5));
out.println("6 - 5 == " + (6-5));
out.println("6 * 5 == " + (6*5));
out.println("6 / 5 == " + (6/5));
```

OUTPUT

```
6 + 5 == 11
6 - 5 == 1
6 * 5 == 30
6 / 5 == 1
```

© A+ Computer Science - www.apluscompsci.com

Math operations can be performed on integers and on decimals.

An integer divided by an integer results in an integer.

```
2/3=0
```

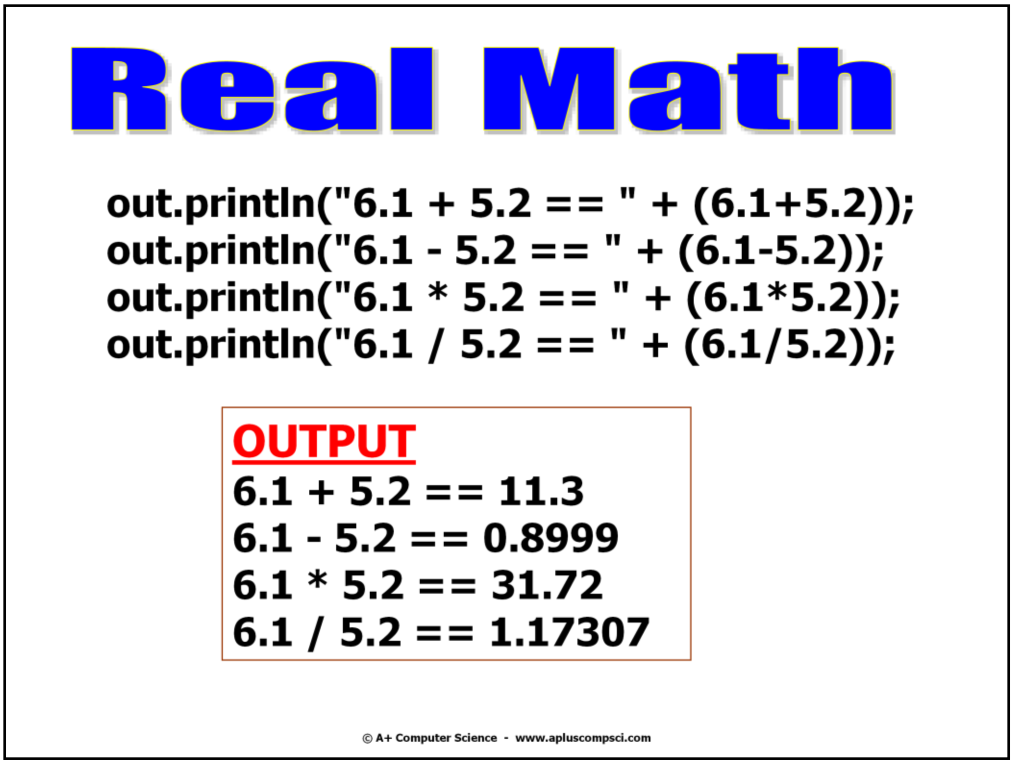
```
3/2=1
```

```
5/4=1
```

```
4/5=0
```

```
7/2=3
```

```
2/7=0
```



Real Math

```
out.println("6.1 + 5.2 == " + (6.1+5.2));
out.println("6.1 - 5.2 == " + (6.1-5.2));
out.println("6.1 * 5.2 == " + (6.1*5.2));
out.println("6.1 / 5.2 == " + (6.1/5.2));
```

OUTPUT

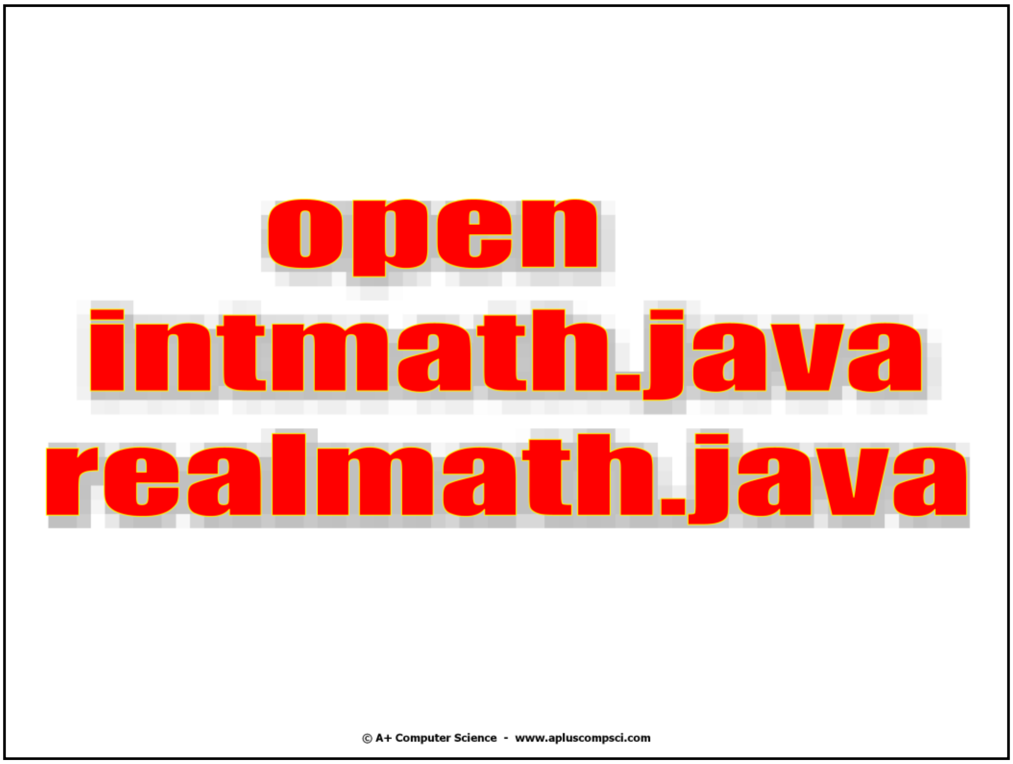
```
6.1 + 5.2 == 11.3
6.1 - 5.2 == 0.8999
6.1 * 5.2 == 31.72
6.1 / 5.2 == 1.17307
```

© A+ Computer Science - www.apluscompsci.com

Math operations can be performed on integers and on decimals.

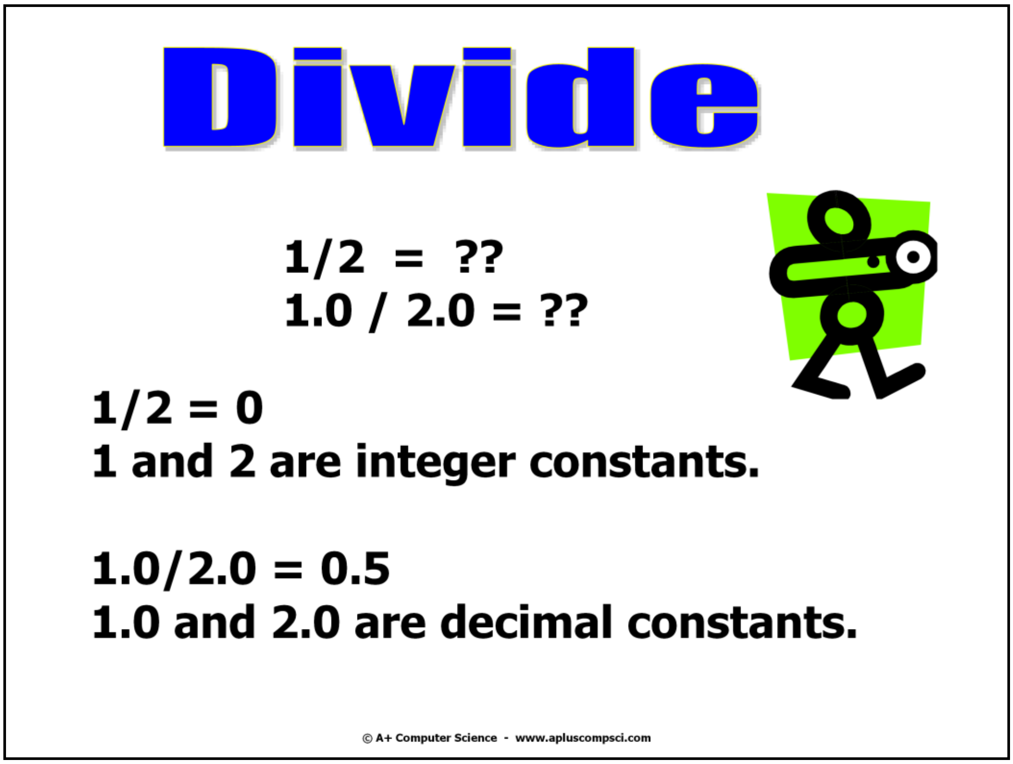
As long as one part of the math is a decimal, the result is a decimal.

```
2.0/3=0.66

3/2.0=1.5

5/4.0=1.25

4.0/5=0.8

7.0/2=3.5

2/7.0=0.2857142
```



open intmath.java realmath.java

© A+ Computer Science - www.apluscompsci.com



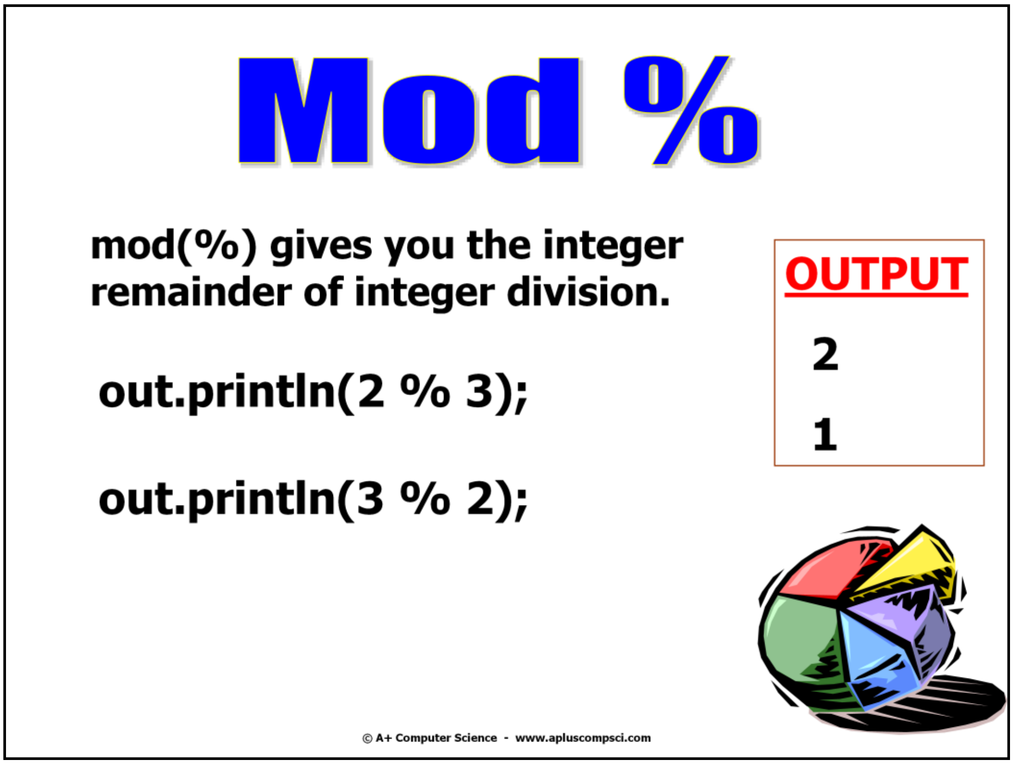
Divide

1/2 = ??
1.0 / 2.0 = ??

1/2 = 0
1 and 2 are integer constants.

1.0/2.0 = 0.5
1.0 and 2.0 are decimal constants.

© A+ Computer Science - www.apluscompsci.com



Mod %

mod(%) gives you the integer remainder of integer division.

```
out.println(2 % 3);

out.println(3 % 2);
```

OUTPUT

```
2
1
```

© A+ Computer Science - www.apluscompsci.com

Modulus is the remainder of division.

$\underline{0}$

3|2

0

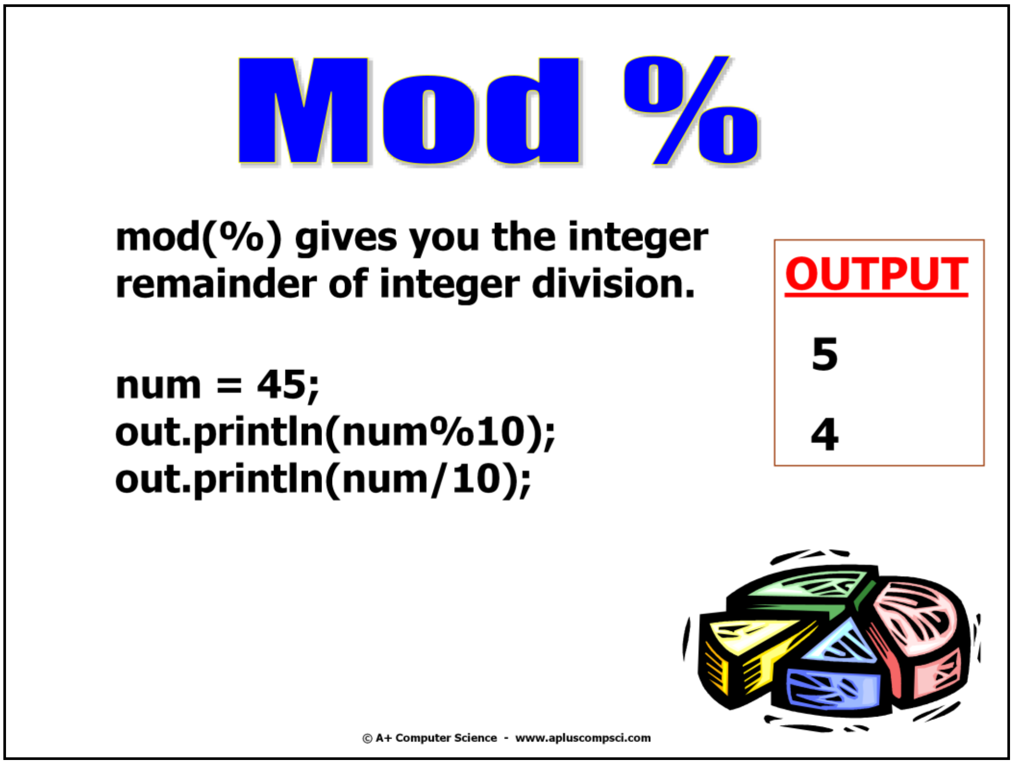
2 is the remainder

$\underline{1}$

2|3

2

1 is the remainder



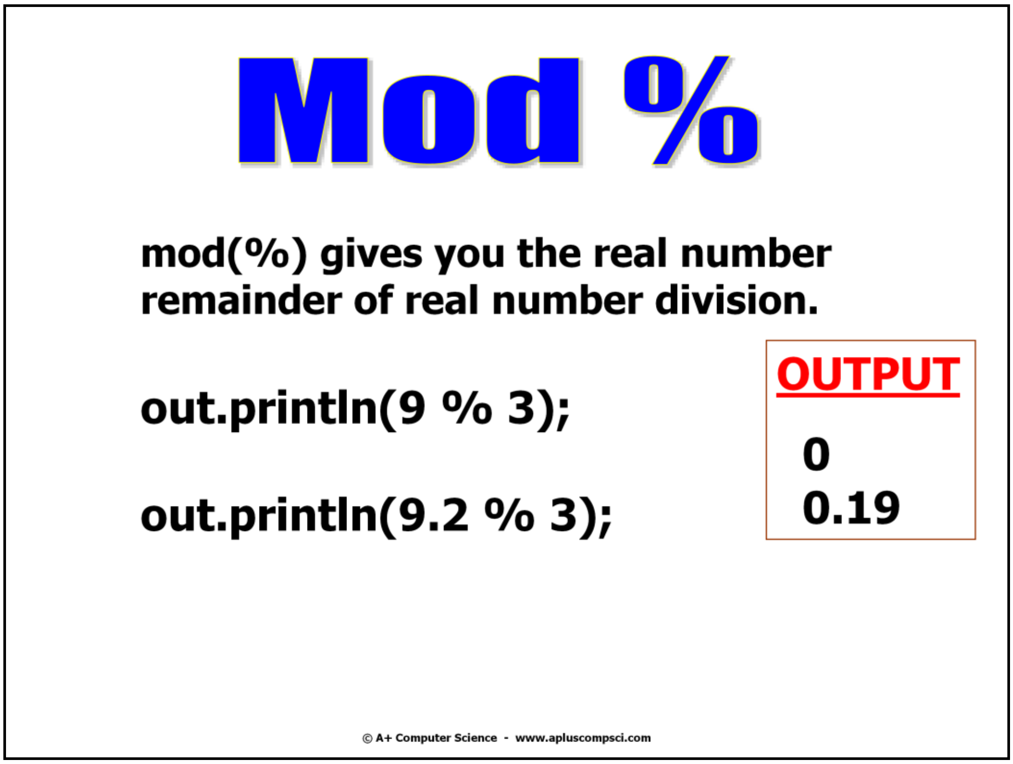
Mod %

mod(%) gives you the integer remainder of integer division.

```
num = 45;
out.println(num%10);
out.println(num/10);
```

OUTPUT

```
5

4
```



Mod %

mod(%) gives you the real number remainder of real number division.

```
out.println(9 % 3);

out.println(9.2 % 3);
```

OUTPUT

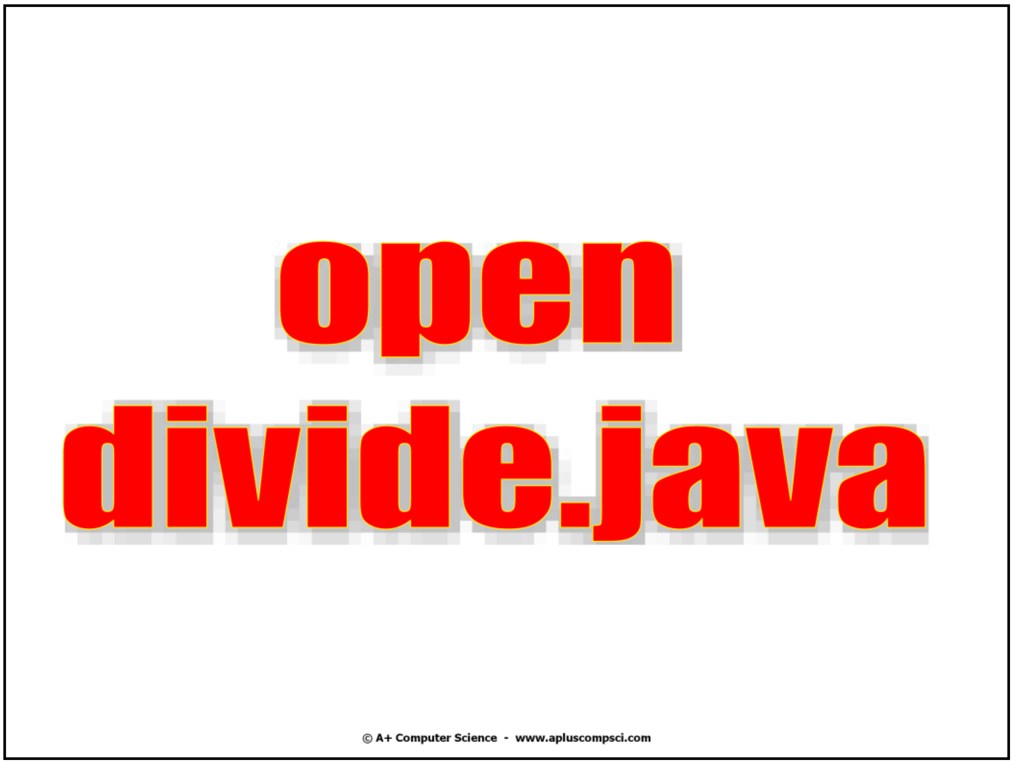
```
0
0.19
```

© A+ Computer Science - www.apluscompsci.com

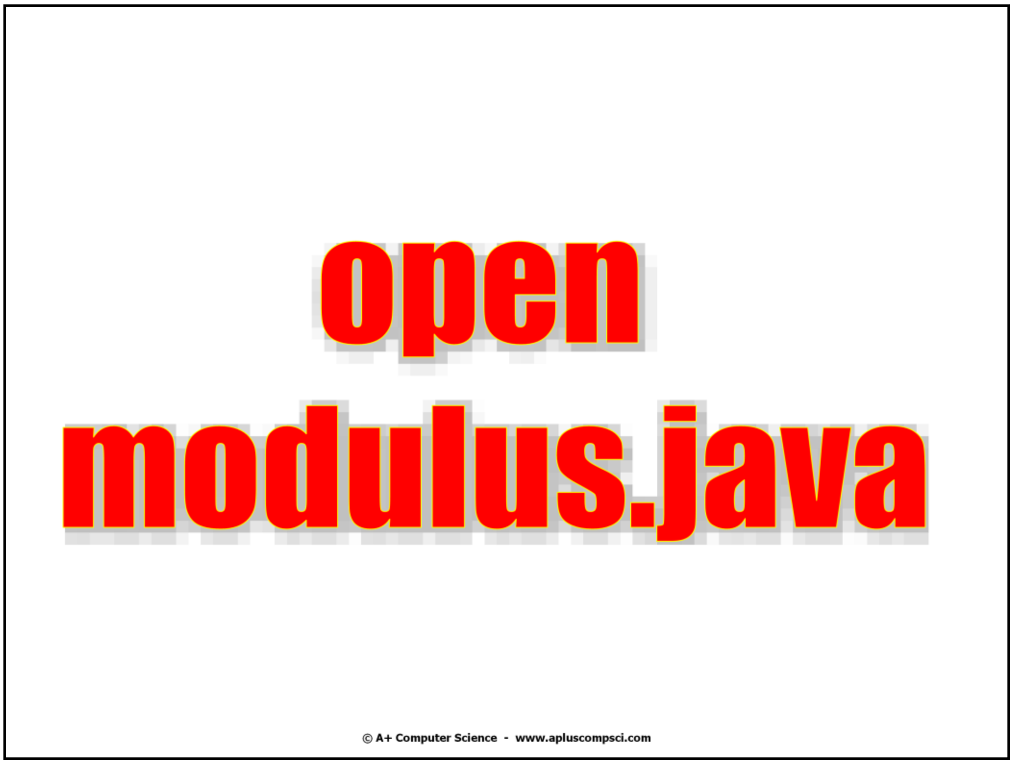
Modulus is the remainder of division.

```
  3
3|9
  9
  0 is the remainder
```

```
  3
3|9.2
  9
  0.2 is the remainder
```

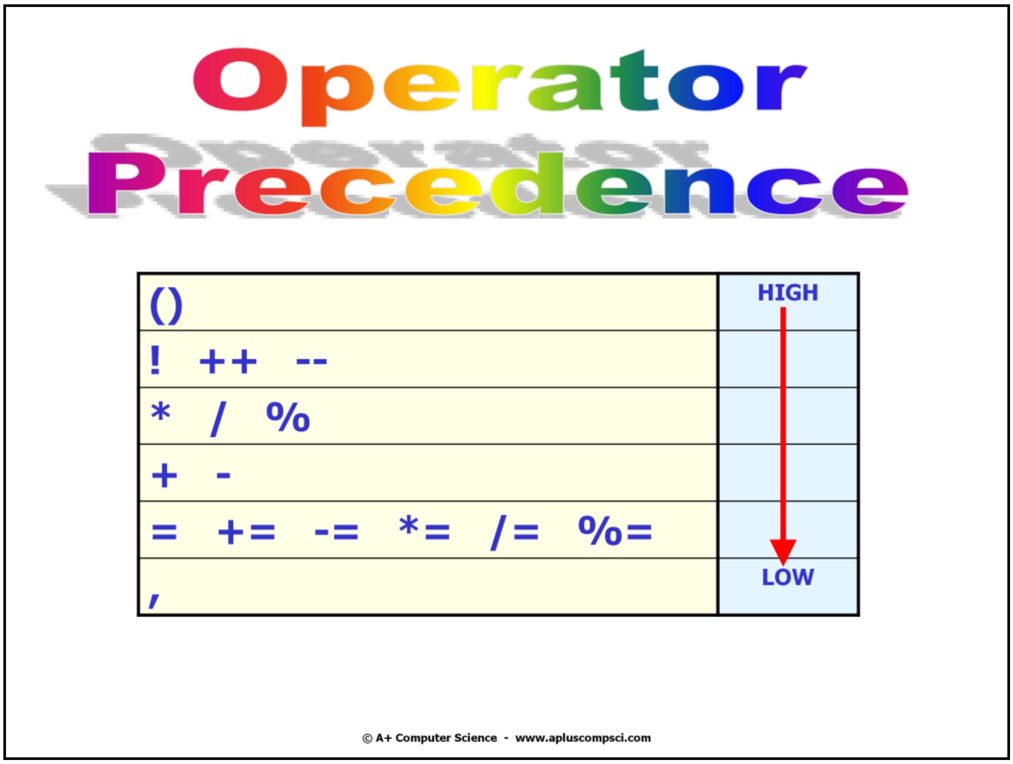


open divide.java



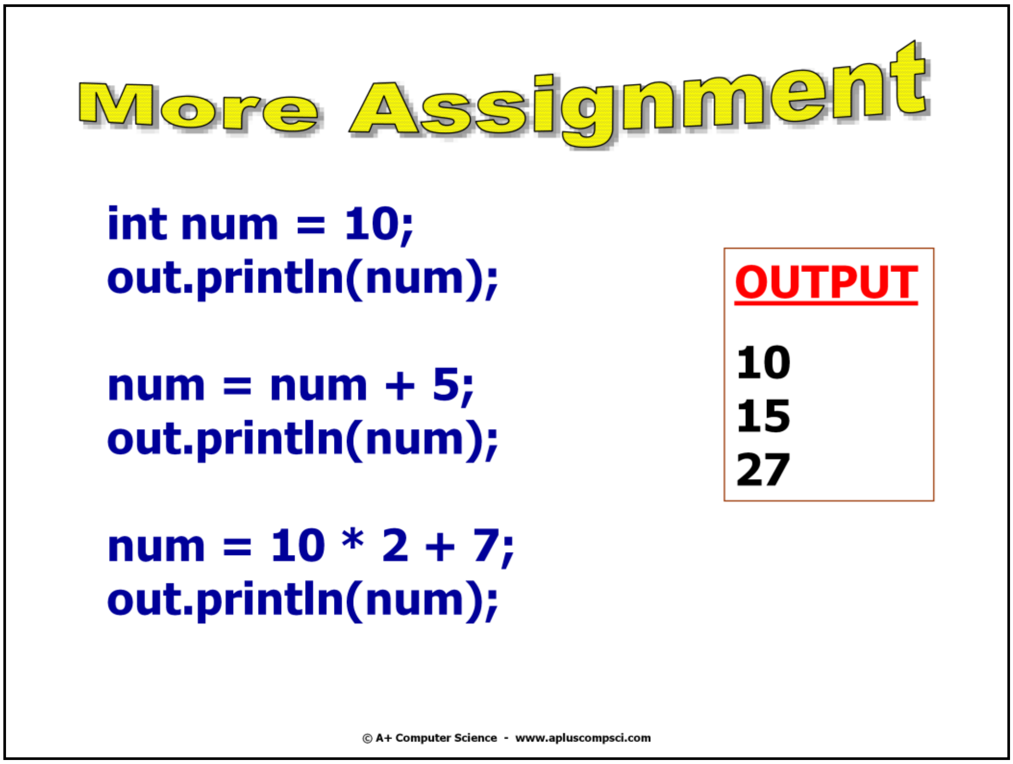
open modulus.java

© A+ Computer Science - www.apluscompsci.com



Operator Precedence

()	HIGH
! ++ --	
* / %	
+ -	
= += -= *= /= %=	
,	LOW



More Assignment

```
int num = 10;
out.println(num);

num = num + 5;
out.println(num);

num = 10 * 2 + 7;
out.println(num);
```

OUTPUT

```
10
15
27
```

© A+ Computer Science - www.apluscompsci.com

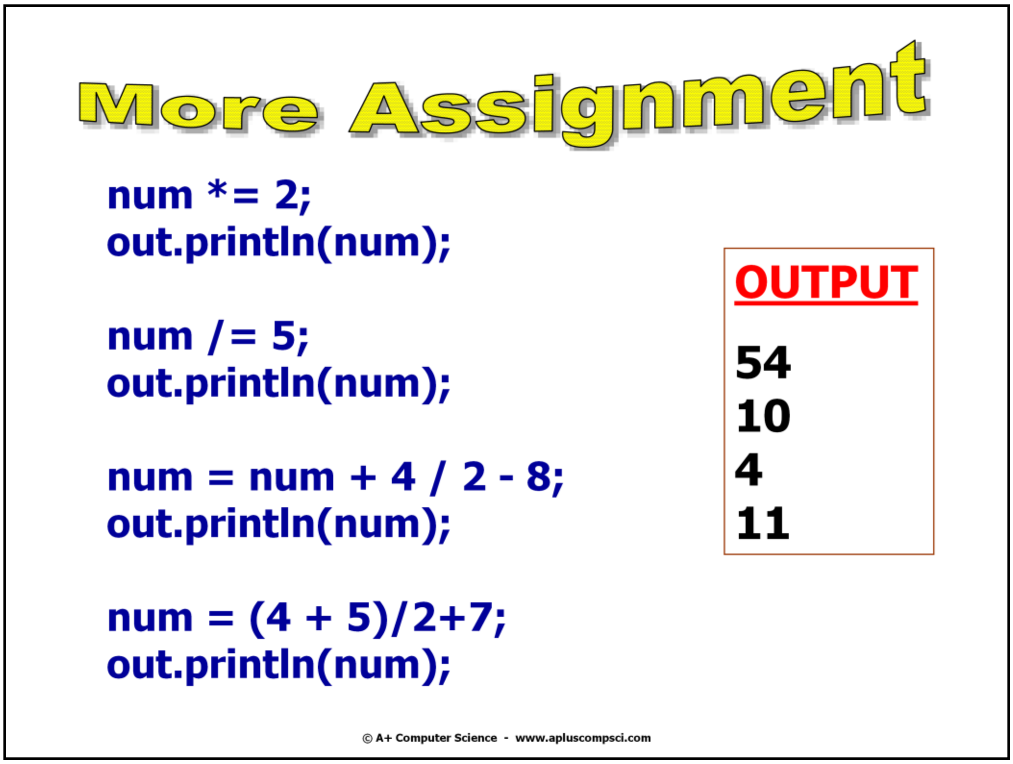
`num` starts out with the value 10.

`num` is then increased by 5. `num = 10+5`

`num` is now 15.

`num` is lastly assigned the value 10*2+7. `num=27`

`*` and `/` have higher precedence than + and –.



More Assignment

```
num *= 2;
out.println(num);

num /= 5;
out.println(num);

num = num + 4 / 2 - 8;
out.println(num);

num = (4 + 5)/2+7;
out.println(num);
```

OUTPUT

```
54
10
4
11
```

© A+ Computer Science - www.apluscompsci.com

`num` starts out with the value 54. `num=27*2`

`*=` is equal to `num=num*2`

`*=` is also the same as `num=num*(int)2`

`*=` auto casts

`num` is now 54.

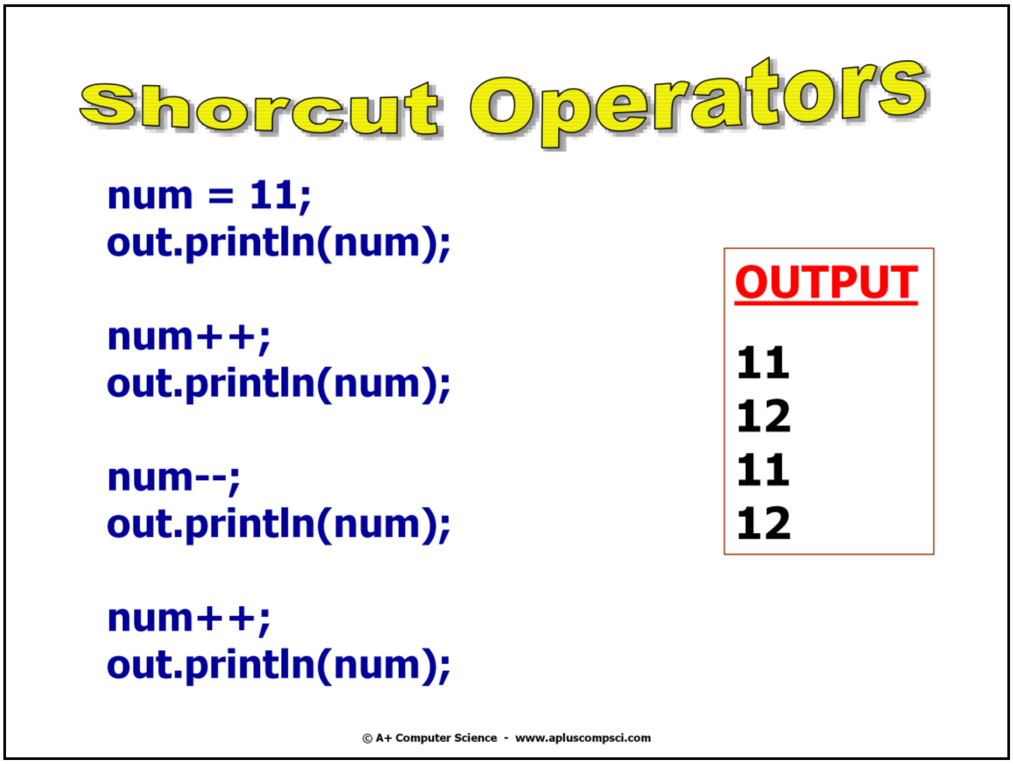
`num` is then divided by 5. `num = 54/5`

`num` is now 10.

`num` is then assigned the value 10+4/2-8. `num = 4`

`num` is then assigned the value (4+5)/2+7. `num = 11`

Parenthesis have higher precedence than math operations.



Shorcut Operators

```
num = 11;
out.println(num);

num++;
out.println(num);

num--;
out.println(num);

num++;
out.println(num);
```

OUTPUT

```
11
12
11
12
```

© A+ Computer Science - www.apluscompsci.com

`num` starts out with the value 11. `num=11`

`++ is the same as num=num+1`

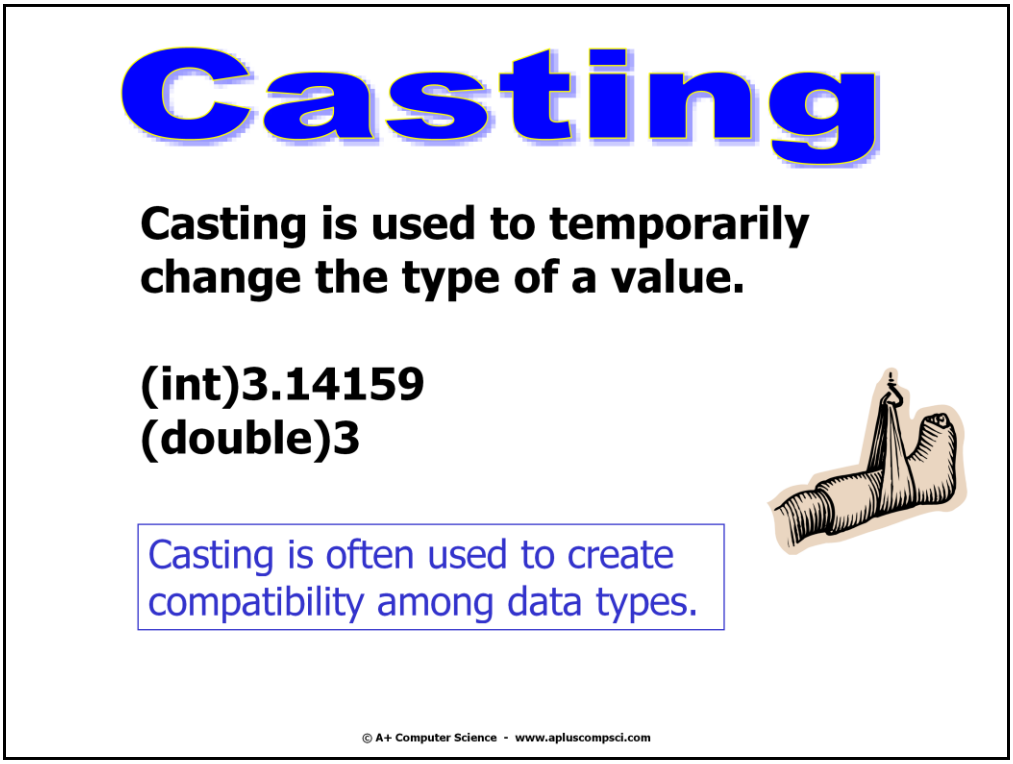
`num` is then increased by 1. `num = 12`

`-- is the same as num=num-1`

`num` is then decreased by 1. `num = 11`

`num` is then increased by 1. `num = 12`

open
assignment.java
shortcuts.java

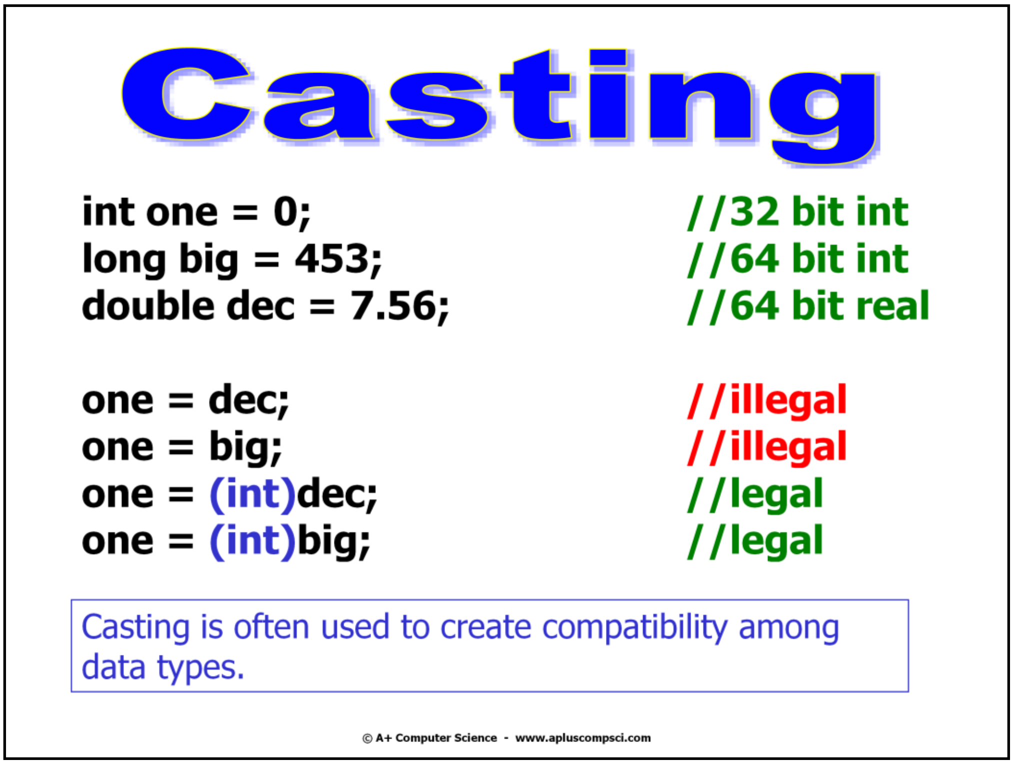


Casting

Casting is used to temporarily change the type of a value.

```
(int)3.14159
(double)3
```

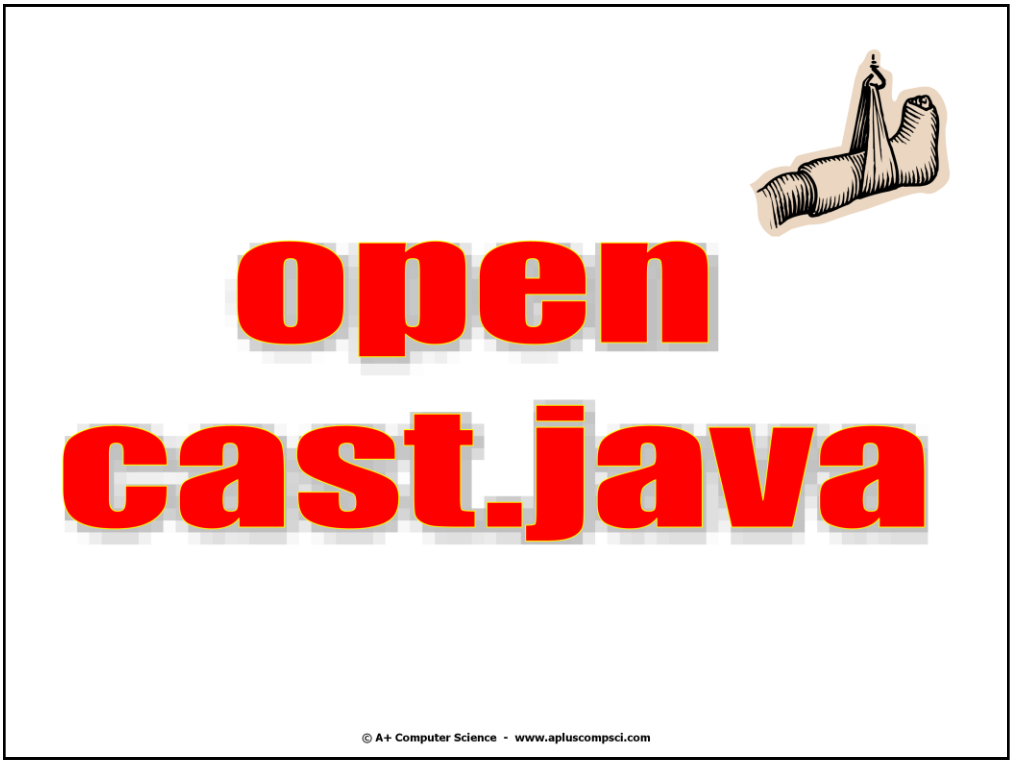
Casting is often used to create compatibility among data types.



Casting

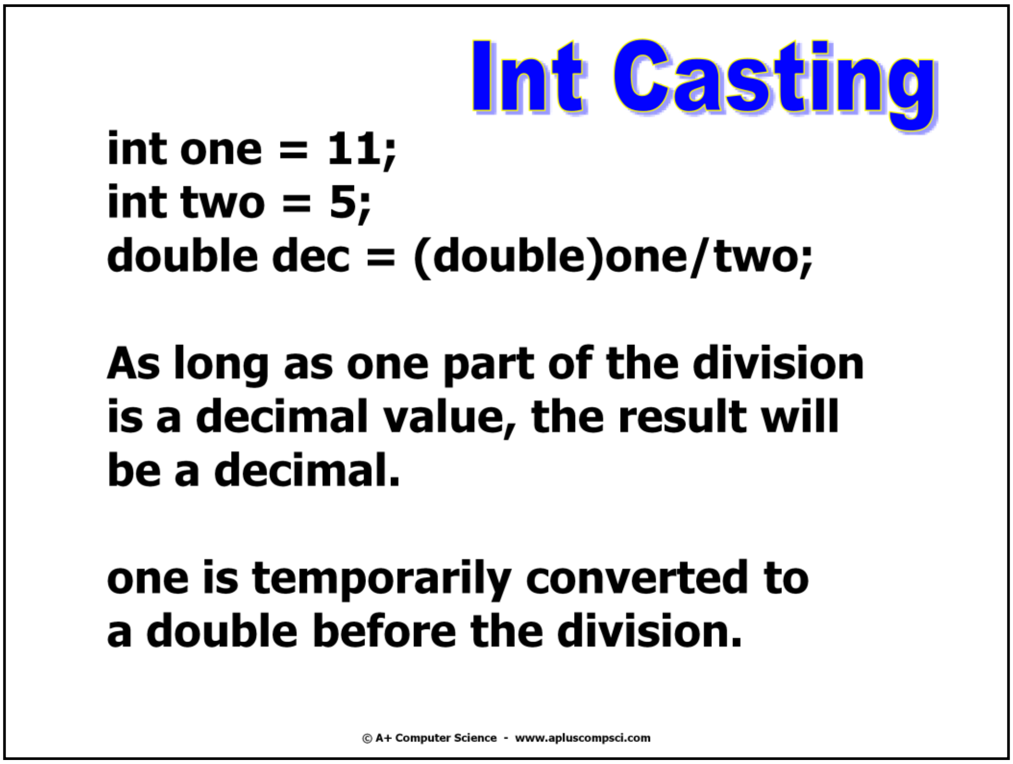
```
int one = 0;                    //32 bit int
long big = 453;                 //64 bit int
double dec = 7.56;              //64 bit real

one = dec;                      //illegal
one = big;                      //illegal
one = (int)dec;                 //legal
one = (int)big;                 //legal
```

Casting is often used to create compatibility among data types.



open cast.java

© A+ Computer Science - www.apluscompsci.com

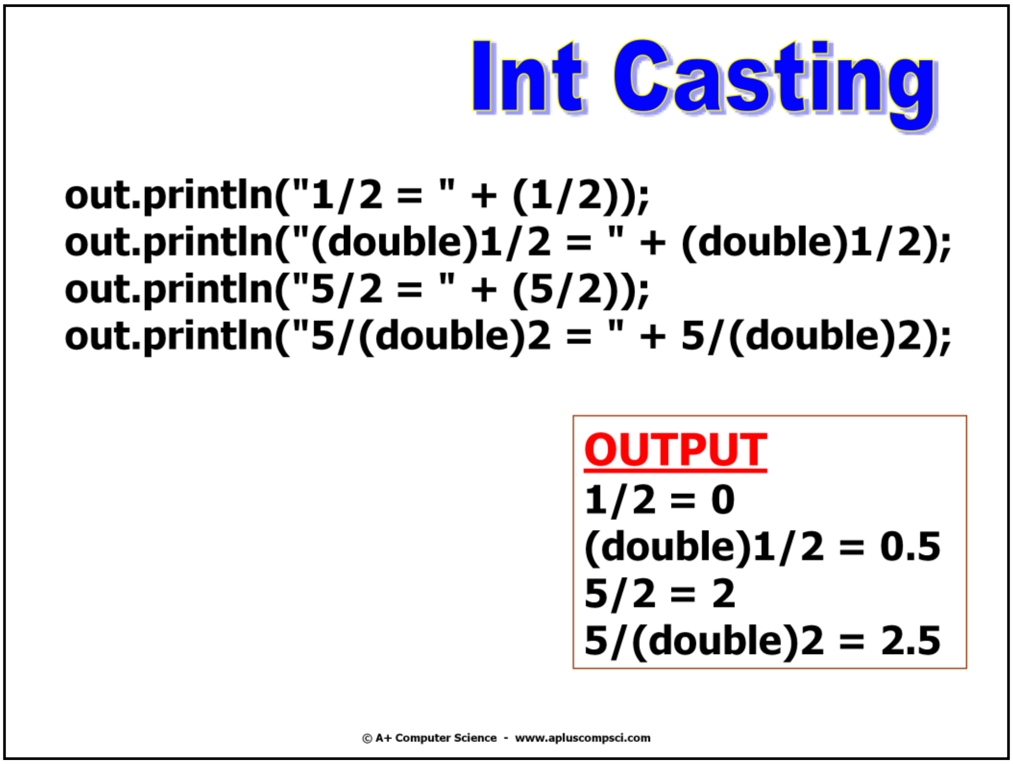


Int Casting

```
int one = 11;
int two = 5;
double dec = (double)one/two;
```

As long as one part of the division is a decimal value, the result will be a decimal.

one is temporarily converted to a double before the division.

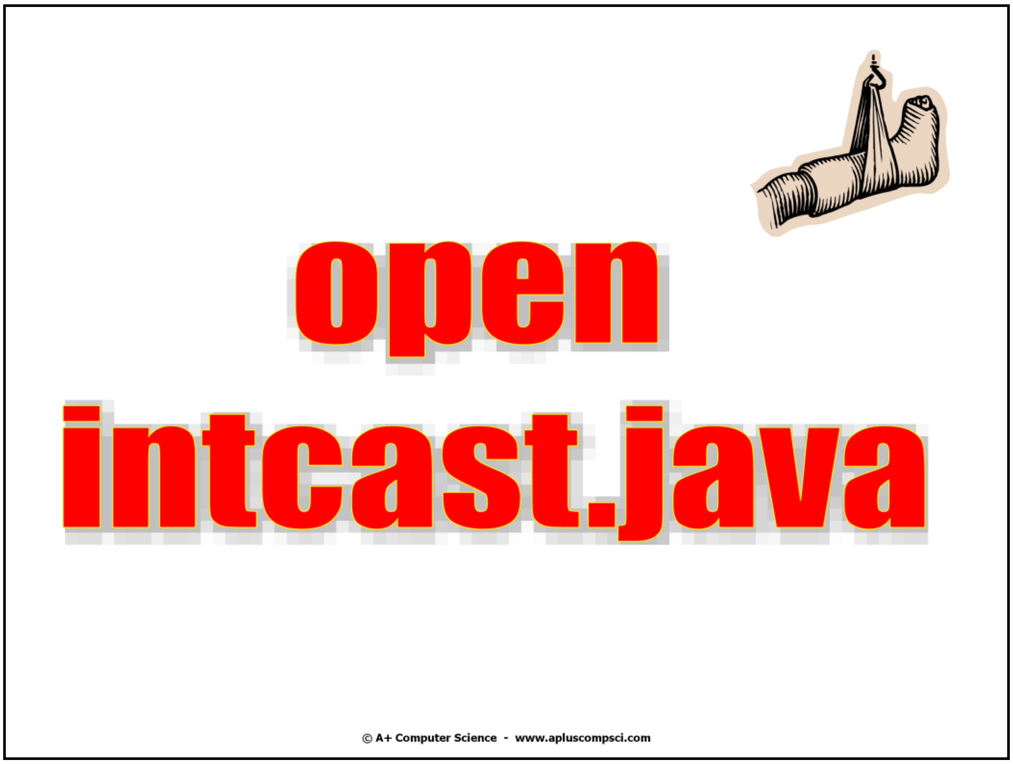


Int Casting

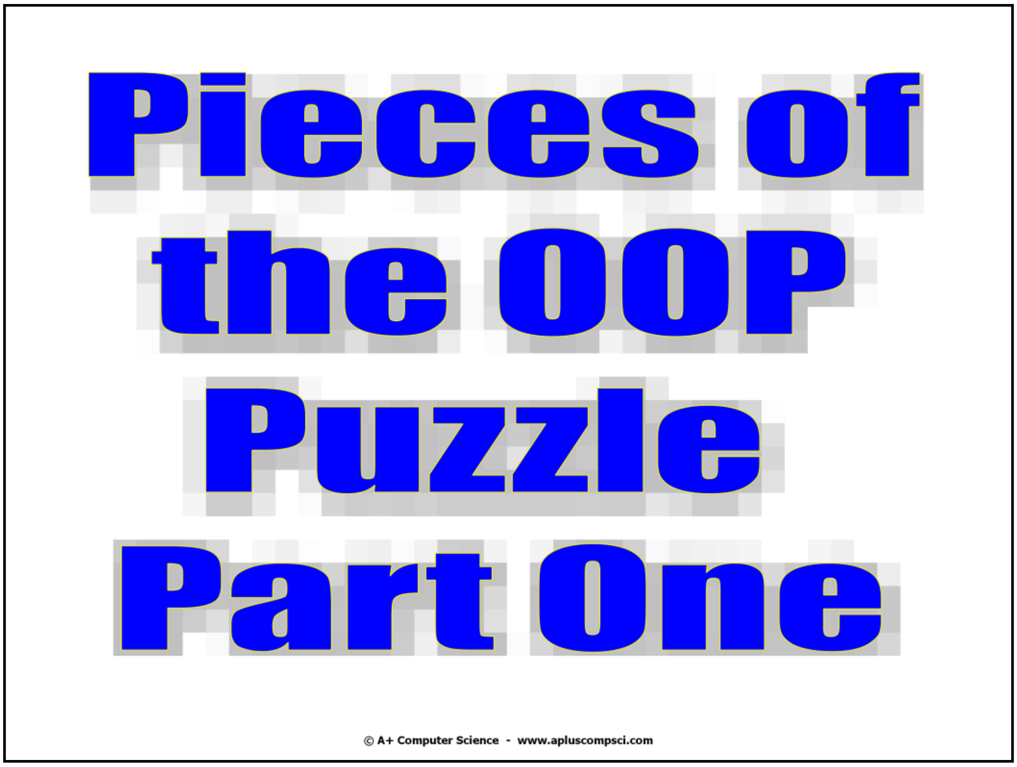
```
out.println("1/2 = " + (1/2));
out.println("(double)1/2 = " + (double)1/2);
out.println("5/2 = " + (5/2));
out.println("5/(double)2 = " + 5/(double)2);
```

OUTPUT

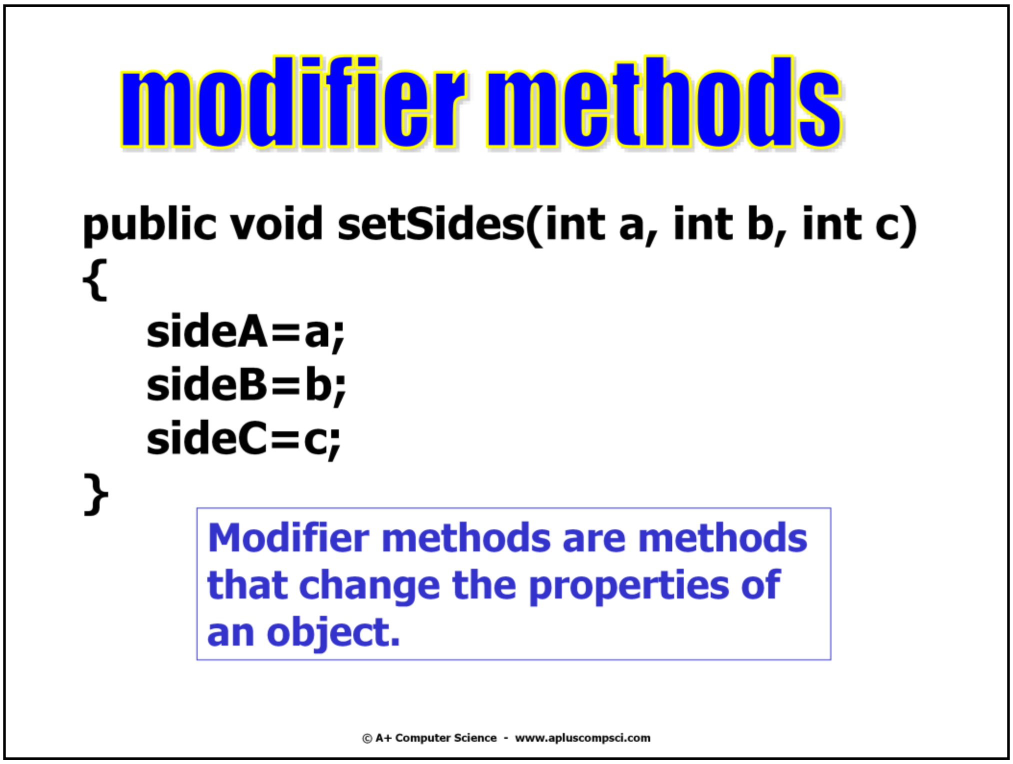
```
1/2 = 0
(double)1/2 = 0.5
5/2 = 2
5/(double)2 = 2.5
```



open intcast.java



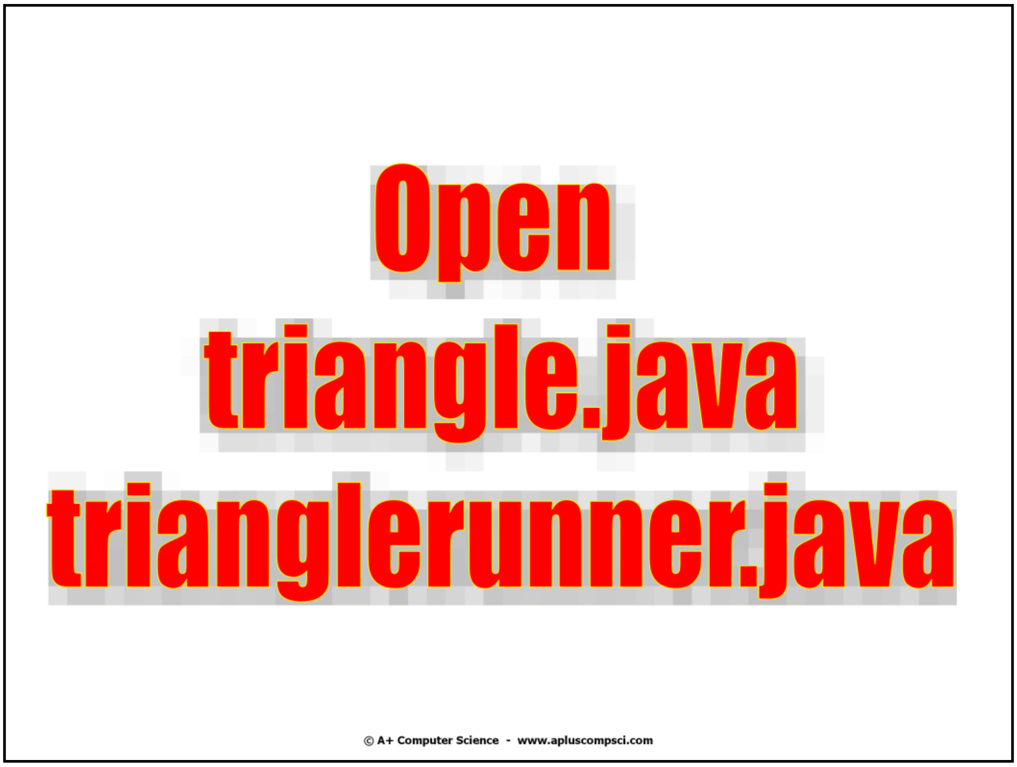
Pieces of the OOP Puzzle Part One



modifier methods

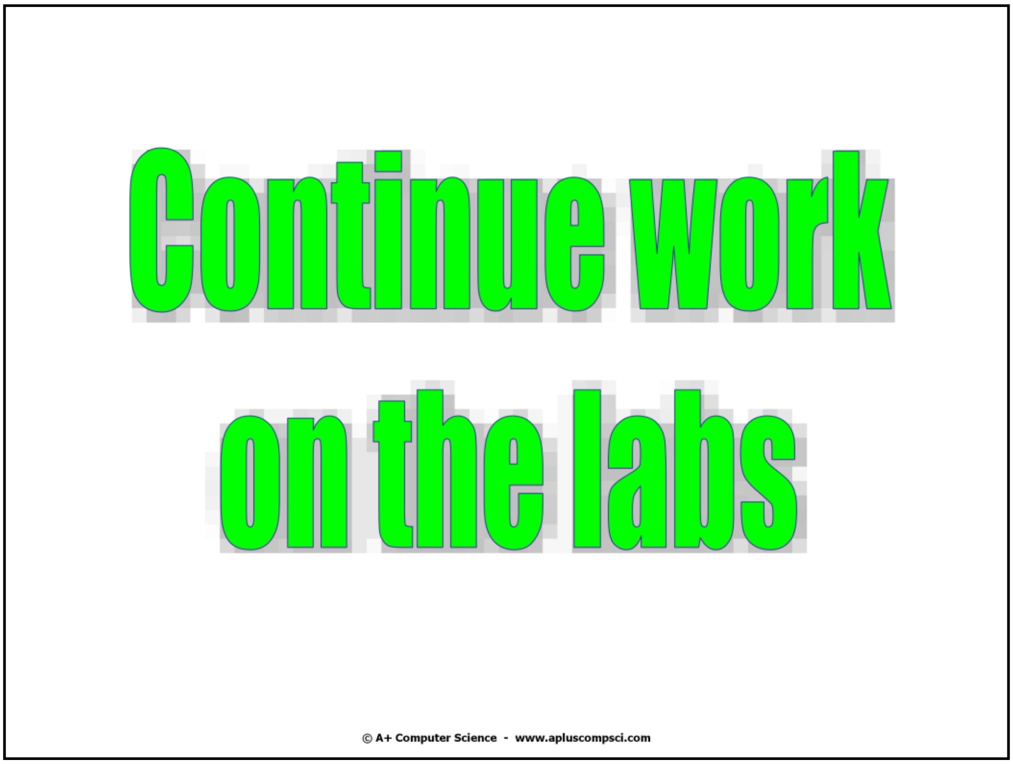
```
public void setSides(int a, int b, int c)
{
   sideA=a;
   sideB=b;
   sideC=c;
}
```

Modifier methods are methods that change the properties of an object.



Open
triangle.java
trianglerunner.java

© A+ Computer Science - www.apluscompsci.com



Continue work on the labs

© A+ Computer Science - www.apluscompsci.com