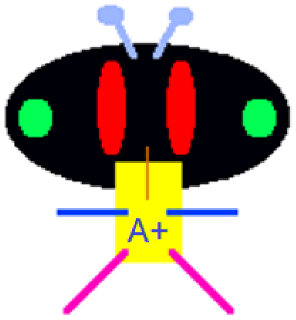
A+ Computer Science

METHODS

Classes

Class

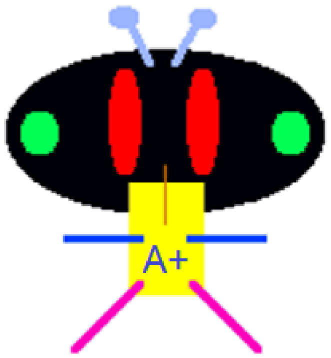
```
public class AplusBug
{
  public void speak()
  {
    out.println("chirp-chirp");
  }
}
```



A class is a blueprint for creating objects.
You can instantiate as many objects as needed.

Instantiation

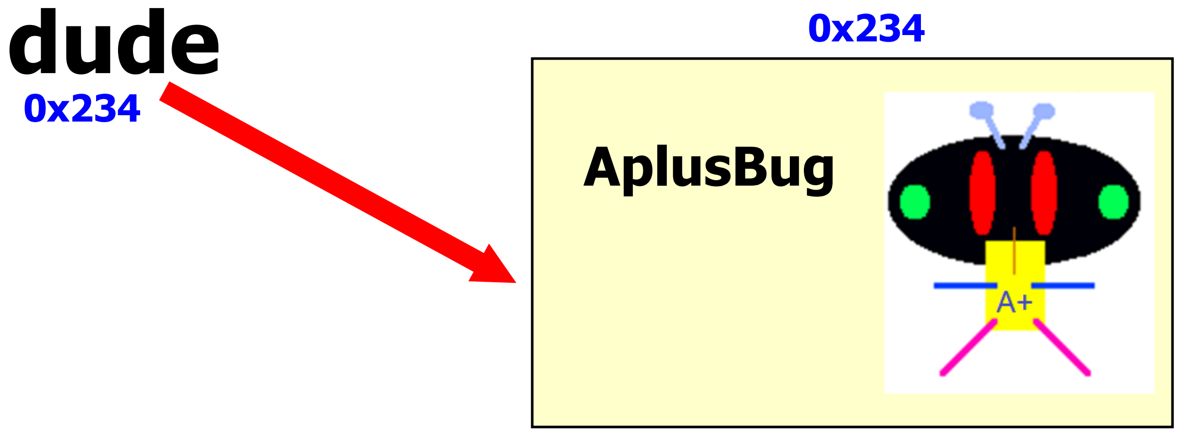
```
AplusBug  dude  =  new  AplusBug();
```



new AplusBug() creates a new AplusBug object.

Instantiation

```
AplusBug dude  = new  AplusBug();
```



dude is a reference variable that refers to an AplusBug object.

Instantiation

```
AplusBug  dude1  =  new  AplusBug();
AplusBug  dude2  =  new  AplusBug();
AplusBug  dude3  =  new  AplusBug();
AplusBug  dude4  =  new  AplusBug();
AplusBug  dude5  =  new  AplusBug();
```

You can make as many as you need.

Methods

What is a method?

A method is a storage location for related program statements. When called, a method usually performs a specific task.

System.out.println( )

Common Methods

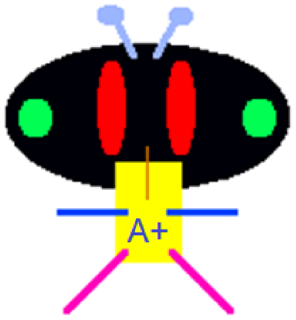
Math.random()

keyboard.nextInt( )

System.out.println( )

Methods

```
public class AplusBug
{
   public void speak()
   {
      out.println("chirp-chirp");
   }
}
```



Methods are defined inside of a class. You can define as many methods as needed.

Defining Methods

```
public void speak()
{
    out.println("chirp-chirp");
}
```

Method speak is public and does not return a value. Method speak contains one line of code that prints out chirp-chirp.

Methods

access | return type | name | params

code

```
public          void        speak(    )
{
  System.out.println("chirp-chirp");
}
```

Method Calls

```
AplusBug dude  = new  AplusBug();
dude.speak();
```

OUTPUT

```
chirp-chirp
```

Once you have instantiated an object, you can call the methods contained in the class.

Method Calls

```
AplusBug dude  = new  AplusBug();
dude.speak();
dude.speak();
dude.speak();
```

OUTPUT

```
chirp-chirp
chirp-chirp
chirp-chirp
```

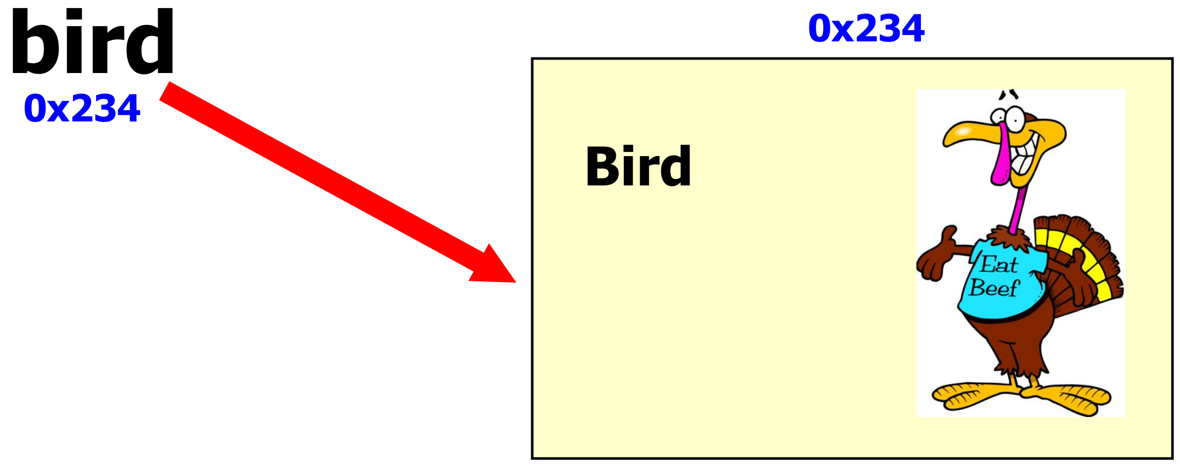
dude can use any of the methods from the AplusBug class as many times as needed.

Basic Bird Class

```
public class Bird
{
   public void speak()
   {
      out.println("chirp-chirp");
   }
   public void sayName()
   {
      out.println("baby bird");
   }
}
```

Instantiation

```
Bird bird = new  Bird();
```



bird is a reference variable that refers to a Bird object.

Instantiation

```
Bird bird = new  Bird();
```

In order to use the Bird class methods, you must instantiate a new Bird object by calling the Bird class constructor.

Instantiation

```
Bird bird = new  Bird();
Bird one = new Bird();
Bird two = new Bird();
Bird three = new Bird();
```

You can create as many new Bird()s as you need.

Instantiation

```
Bird bird = new  Bird();
bird.sayName();
bird.sayName();
```

OUTPUT

```
baby bird
baby bird
```

Once you have a reference to a Bird, you can call the methods that belong to the Bird class.

Bird Runner

```
//Code in the Bird Runner
Bird bird = new Bird();
bird.speak();
bird.sayName();
bird.speak();
bird.sayName();
bird.speak();
```

OUTPUT

```
chirp-chirp
baby bird
chirp-chirp
baby bird
chirp-chirp
```

Bird.java
Birdrunner.java

Return Methods

Return Methods

access return type name params

code

Return Methdos

```
public class Fun
{
   public int times(int num1, int num2)
   {
      return num1*num2;
   }
}
```

Return methods are defined inside of a class in the same way you define void methods.

Return

```
public int times(  int num1,   int num2 )
{
    return num1*num2;
}
```

The reserved word return is used inside of a method when a value needs to be sent back to the method call. The value sent back must match the return type.

Return Methods

```
public class Fun
{
  public int times(int num1, int num2)
  {
    return num1*num2;
  }
}

Fun aplus = new Fun();
System.out.println( aplus.times( 4, 5 ) );
```

Return Methods

```
public class Fun
{
  public int times(int num1, int num2)
  {
    return num1*num2;
  }
}

Fun aplus = new Fun();
int storeIt = aplus.times( 4, 5 );
```

Defining Parameters

```
public int times(  int num1,   int num2 )
{
   return 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.

Formal Parameters

```
public double fun(  int x,  double y )
{
   return x*y-x;
}
```

Formal parameters are defined with types as part of the method signature. Methods can have as many parameters as needed.

Actual Parameters

```
public int times(  int num1,   int num2 )
{
    return num1*num2;
}

System.out.println( aplus.times(3 , 5) );
```

Actual parameters are the parameters in the method call.
Actual parameters can be primitive values or references.

fun.java
funrunner.java

Work on Programs!

Crank Some Code!

Static Methods

```
public class Fun2
{
   public static int times(int num1, int num2)
   {
      return num1*num2;
   }
}
```

Static return methods are defined inside of a class in the same way you define non-static methods.

Static Methods

```
public static int times(  int num1,   int num2 )
{
    return num1*num2;
}

System.out.println( Fun2.times(3 , 5) );
```

The word static can be placed on a method before the return type to make a method that can be called without an object instantiation.

Static Methods

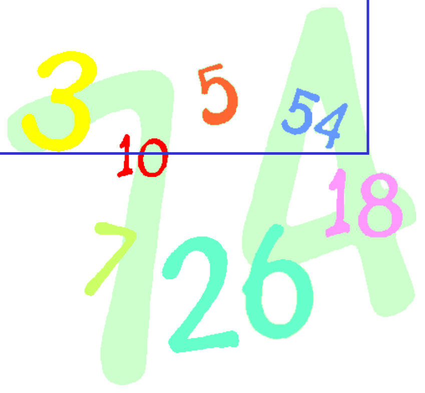
```
public static int times(  int num1,   int num2 )
{
    return num1*num2;
}

System.out.println( Fun2.times(3 , 5) );
```

Static methods can be called directly on the class name.

Static Methods

```
out.println(Math.floor(3.254));
out.println(Math.ceil(2.45));
out.println(Math.pow(2,7));
```

All of the Math methods are static as they are called directly on the class name.



Static Methods

```
public static int times(  int num1,   int num2 )
{
    return num1*num2;
}

Fun2 aplus = new Fun2();
System.out.println( aplus.times(3 , 5) );
```

Static methods can still be called on a reference.

fun2.java
fun2runner.java

Math Return Methods

Math
frequently used methods

Name	Use
floor(x)	rounds x down
ceil(x)	rounds x up
pow(x,y)	returns x to the power of y
abs(x)	returns the absolute value of x
sqrt(x)	returns the square root of x

part of java.lang package

Math frequently used methods	
Name	**Use**
round(x)	rounds x to the nearest whole number
min(x,y)	returns smallest of x and y
max(x,y)	returns biggest of x and y
random()	returns a double >=0.0 and < 1.0

part of java.lang package

Math Methods

```
Scanner keyboard =
                 new Scanner(System.in);

double num = keyboard.nextDouble();
out.println(Math.ceil(num));
```

INPUT
3.45

OUTPUT
4.0

num
3.45

return methods

Math Methods

```
out.println(Math.floor(3.254));
out.println(Math.ceil(2.45));
out.println(Math.pow(2,7));
out.println(Math.abs(-9));
out.println(Math.sqrt(256));
```

OUTPUT

```
3.0
3.0
128.0
9
16.0
```

Math Methods

```
out.println(Math.sqrt(144));
out.println(Math.round(3.6));
out.println(Math.max(5,7));
out.println(Math.max(5,-7));
out.println(Math.min(5,7));
out.println(Math.min(5,-7));
```

OUTPUT

```
12.0
4
7
5
5
-7
```

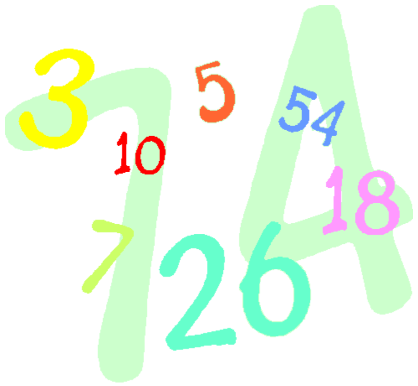
Math Methods

```
out.println(Math.random());
```

OUTPUT

```
0.256
```

random() returns a double in the range 0.0 to 1.0, not including 1.0.



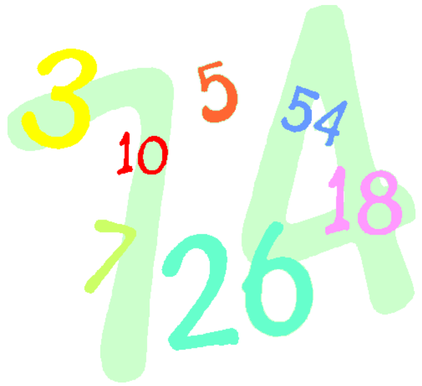
Math Methods

```
out.println(Math.random()*10);
int num = (int)(Math.random()*10);
out.println(num);
```

OUTPUT

```
7.564
4
```

random() returns a double in the range 0.0 to 1.0, not including 1.0.



mathmethods.java
randomone.java

Just For Fun

Formatting Numbers

Formatting Output

How to format

What to format

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

OUTPUT

```
9.24
```

Formatting Output

```
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
```

Formatting Output

Column size

# of decimals

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

type of data

OUTPUT

```
8.26
```

Formatting Output

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

OUTPUT

```
5.342
       5.342
5.342  x
```

realformatone.java
realformattwo.java

Formatting Output

```java
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
```

Formatting Output

```
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
```

intformatone.java
intformattwo.java

Work on Programs!

Crank Some Code!

A+ Computer Science

METHODS