A+ Computer Science

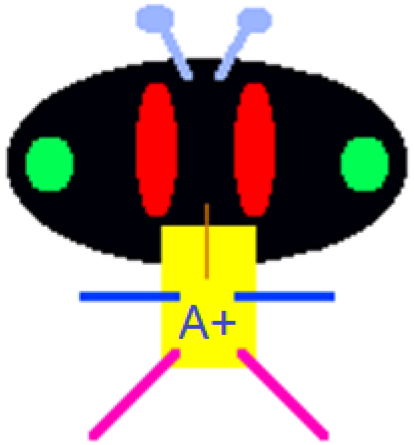
# Class Basics

## Objects

# Object Instantiation

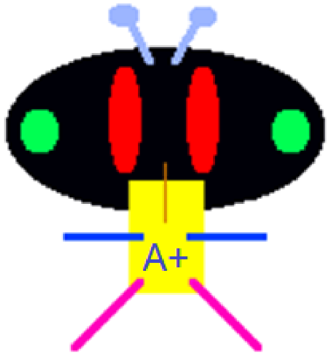
```
new Scanner(System.in);

new AplusBug();
```



# Object Instantiation

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

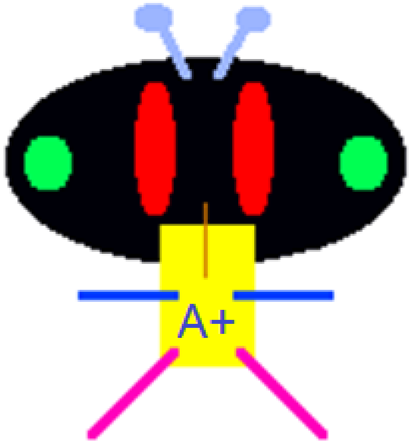


**new AplusBug() creates a new AplusBug object.**

# Object Instantiation

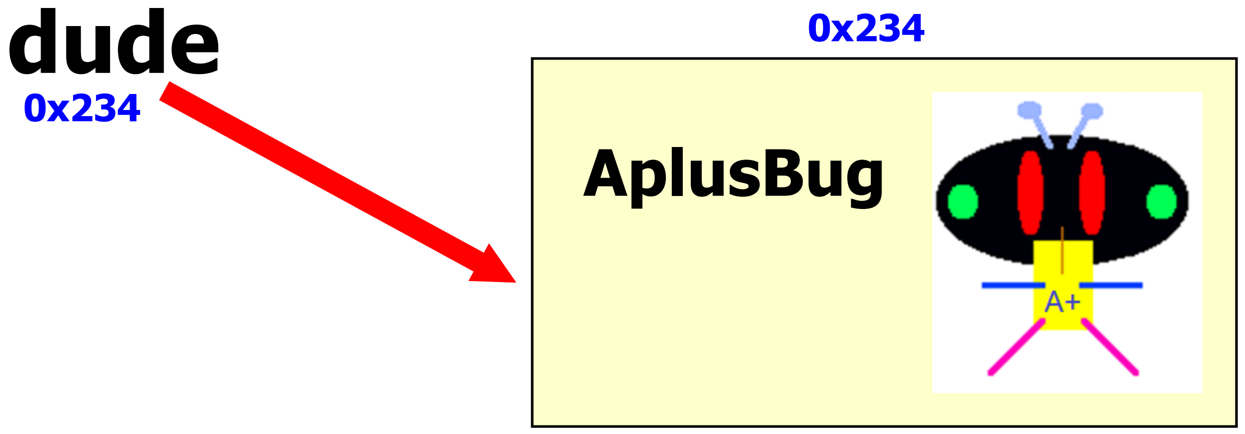
```
Scanner keyboard =
                new Scanner(System.in);

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



# Object Instantiation

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



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

## Instance Variables

# Instance Variables

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

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

## Instance Variables

```
public class Chicken
{
   private int size;
   private boolean fly;


   //other stuff
  //not shown

}
```

Instance variables are shared by all methods in a class.

They are initialized in the constructor.

## Instance Variables

**Each object instantiation has its own set of instance variables.**

```
Chicken a = new Chicken( 9, true );

Chicken b = new Chicken( 33, false );
```

## Constructors

## Constructors

```
public Chicken( int s, boolean c )
{
   size = s;
   fly = c;
}
```

**Constructors are similar to methods. Constructors set the properties of an object to an initial state.**

## Constructors

```
public Chicken( int s, boolean c )
{
   size = s;
   fly = c;
}
```

Constructors often have parameters. The parameters allow data to be passed into the class so that it can be assigned to the instance variables.

## Constructors

```
public class Chicken
{
   private int size;
   private boolean fly;

   public Chicken( int s, boolean c )
   {
         size = s;
         fly = c;
   }
}
```

```
Chicken bird = new Chicken(5,false);
```

## Constructors

```
public class Chicken
{
   private int size;
   private boolean fly;

   public Chicken( int s, boolean c )
   {
         size = s;
         fly = c;
   }
}
```

**Constructors always have the same name as the class.**

chicken.java
chickenrunner.java

Computer Science
www.apluscompsci.com

## Mutator & Accessor Methods

## Mutator Methods

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

# Mutator Methods

```
public class Turkey
{
   private int size;
   private boolean fly;

   public void changeFly( boolean c )
   {
       fly = c;
   }
}
```

void method

mutator method

# mutator Methods

```
public void changeFly( boolean c )
{
     fly = c;
}
```

**Mutator methods are methods that change the properties of an object.**

# Accessor Methods

Accessor methods are public methods that access the private data from a class. They typically return the values of the data.

# Accessor Methods

```
public class Turkey
{
   private int size;
   private boolean fly;

   public boolean canFly()
   {
        return fly;
   }
```

return type

return method

accessor method

## turkey.java
## turkeyrunner.java

# Some Java Provided Classes

## Some Java Classes

Java provides what are called wrapper classes for each of its primitive data types. These wrapper classes can be useful in certain situations.

# Box / Unbox

| primitive | object |
|---|---|
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |
| == | .equals() |

## Box / Unbox

**When assigning a primitive value to a wrapper class, Java automatically calls the wrapper class constructor.**

```
Integer numOne = 99;
Integer numTwo = new Integer(99);
```

```
=99;
=new Integer(99);
```

**These two lines are equivalent.**

# Box / Unbox

When assigning a primitive value to a wrapper class, Java automatically calls the wrapper class constructor.

```
Double numOne = 99.1;
Double numTwo = new Double(99.1);

=99.1;
=new Double(99.1);
```

These two lines are equivalent.

## Box / Unbox

Java will unwrap the primitive values.

```
Integer num = new Integer(3);
int prim = num.intValue();
out.println(prim);
prim = num;
out.println(prim);
```

**OUTPUT**

```
3
3
```

```
prim=num.intValue();
prim=num;
```

**These two lines are equivalent.**

## Box / Unbox

```
Double dub = 9.3;
double prim = dub;
out.println(prim);

int num = 12;
Integer big = num;
out.println(big.compareTo(12));
out.println(big.compareTo(17));
out.println(big.compareTo(10));
```

**OUTPUT**

```
9.3
0
-1
1
```

## Box / Unbox

**Using wrapper classes will make way more sense once when we use the ArrayList class.**

**It does not hurt to learn about this now, but it will make more sense in the context of working with an ArrayList.**

## autoboxunbox.java

Computer Science
www.apluscompsci.com

Work on Programs!

Crank Some Code!

A+ Computer Science

# CLASS BASICS