A+ Computer Science

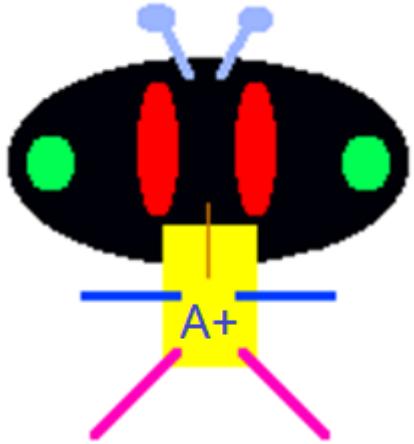
Writing Classes

Objects

Object Instantiation

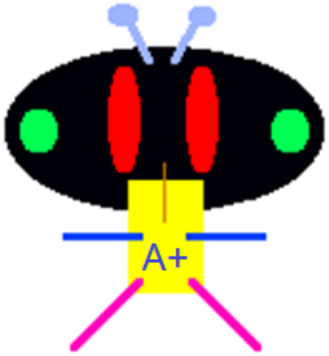
new Scanner(System.in);

new AplusBug();



Object Instantiation

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

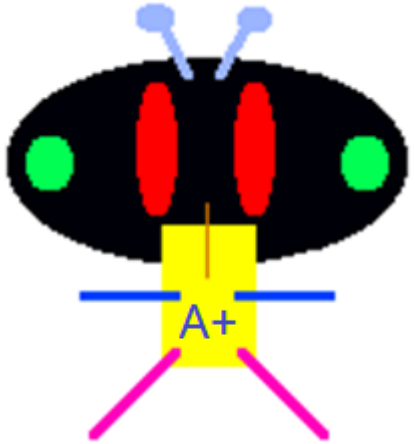


new AplusBug() creates a **new** AplusBug object.

Object Instantiation

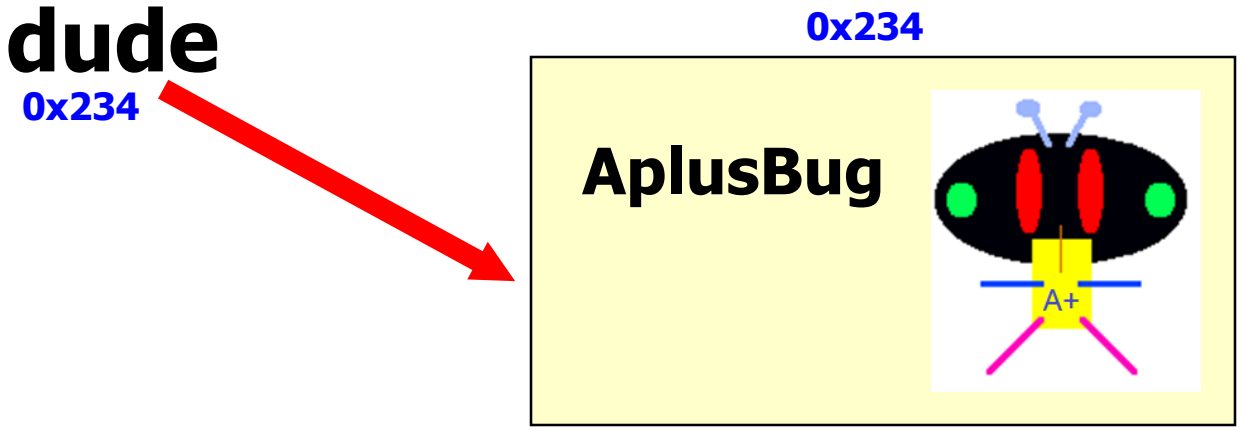
```
Scanner keyboard =
                new Scanner(System.in);
```

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



Object Instantiation

AplusBug dude = new AplusBug();



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

Private
Public

Public

All members with public access can be accessed or modified inside or outside of the class where they are defined.

Private

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

Encapsulation

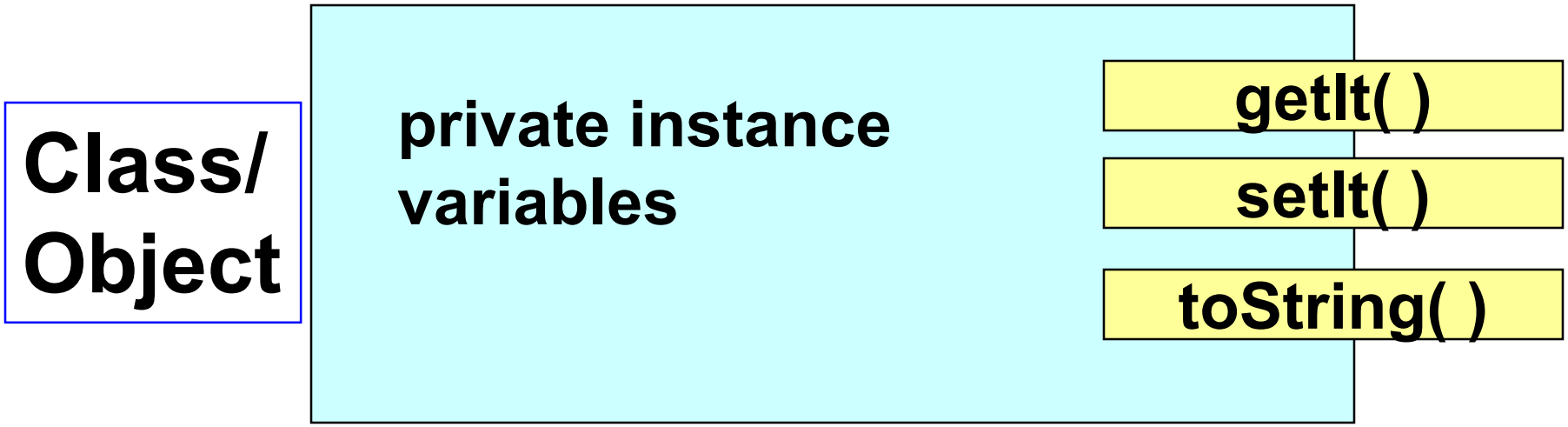
All instance variables should have private access. A set of public methods should be provided to manipulate the private data.

Encapsulation

All data members should have private access. The public constructors, accessor methods, and mutator methods should be used to manipulate the data. All data is tucked away nicely inside the class.

Encapsulation

The public methods give you access to an object's private Instance variables.



frog.java
frogrunner.java

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.

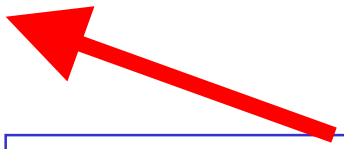
Instance Variables

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


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

   public int getAnswer(){
      return answer;
   }
}
```

Instance variables are shared by all methods in a class.

Instance Variables

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

   //other stuff
   //not shown

}
```



Instance variables are shared by all methods in a class.

They are initialized in the constructor.

Instance Variables

An object "has-a" set of instance variables. The state of an object is based on the value of its instance variables.

The relationship between an object and its instance variables is a "has-a" relationship.

Instance Variables

Each object instantiation has its own set of instance variables.

```
Calc a = new Calc( 5, 7 );

Calc b = new Calc( 33, 6 );
```

calc.java
calcrunner.java

Constructors

Constructors

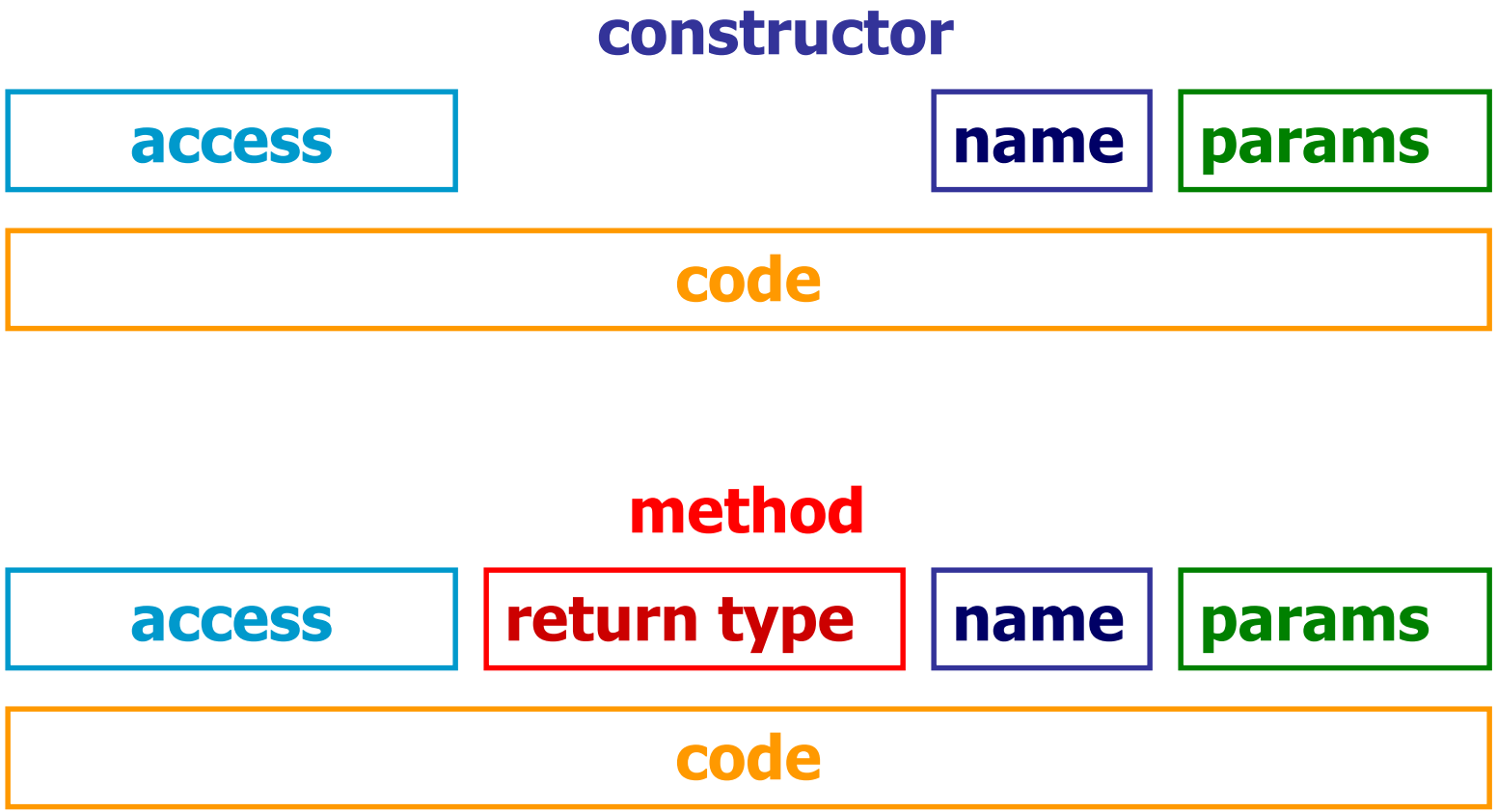
very similar to methods

have the same name as the class

have no return type – no void,int,etc.

initialize all instance variables

Constructors vs. Methods



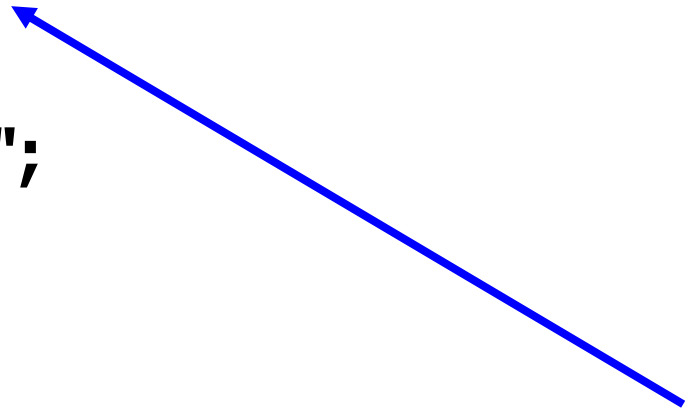
No Argument Constructor

```
public Monster()
{
   name = "scary";
   size = 8.334;
}
```

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

No Argument Constructor

```
public class Monster
{
   private String name;
   private double size;

   public Monster()
   {
      name = "scary";
      size = 8.334;
   }
}
```



```
Monster ghoul = new Monster();
```

Default Constructor

If you do not provide / define any constructors in a class, Java will provide one default constructor (no argument constructor) that initializes all instance variables to zero values.

monster.java
monsterrunner.java

Constructor Parameters

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

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

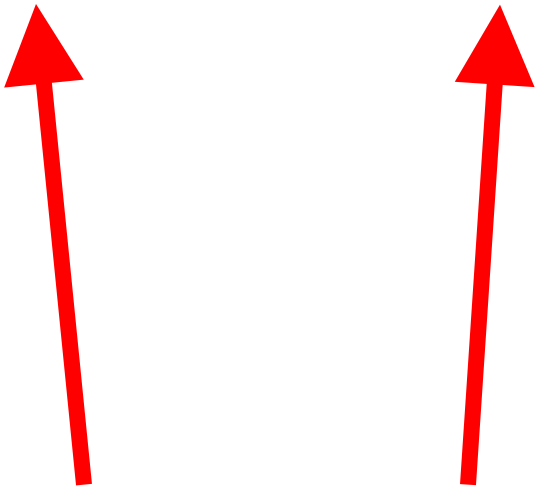
Constructor Parameters

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

Formal parameters are the parameters listed in the signature.

Constructor Parameters

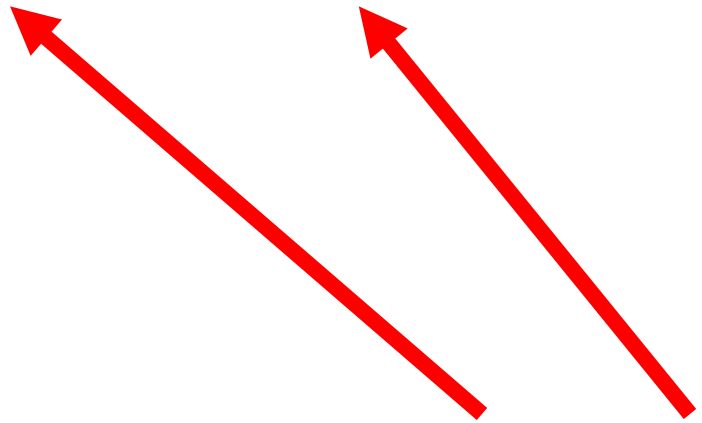
```
public Turkey( int s, boolean c )
{
   size = s;
   fly = c;
}
```



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

Constructor Parameters

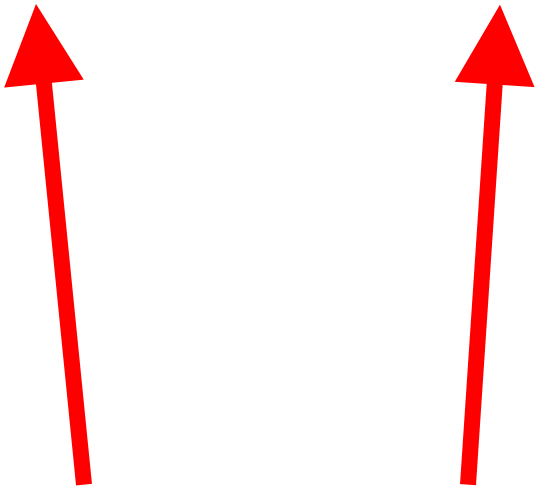
```
public class Turkey
{
   private int size;
   private boolean fly;

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



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

Formal Parameters

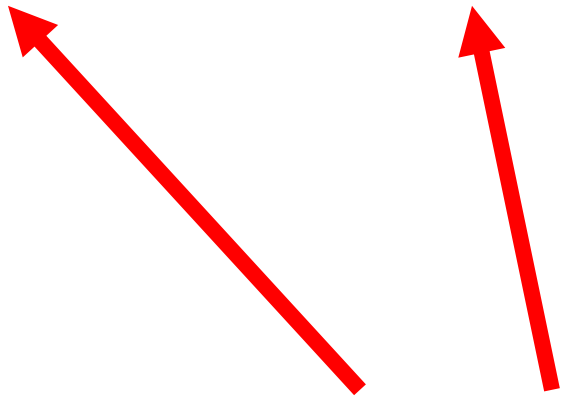
```
public Turkey( int s, boolean c )
{
   size = s;
   fly = c;
}
```



Formal Parameters are the parameters defined as part of the signature. Formal parameters have types and names listed.

Actual Parameters

```
public Turkey( int s, boolean c )
{
   size = s;
   fly = c;
}
Turkey bird = new Turkey(5,false);
```



Actual Parameters are the values being passed to the formal parameters. The values passed in can be constants, primitives, or references.

Mutator & Accessor Methods

Accessor Methods

Accessor methods are public methods that access the private data from a class. Accessors return a single value.

Accessor Methods

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

   public boolean canFly()
   {
      return size < 25;
   }
}
```

return type

return method

accessor method

Accessor Methods

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

   public String toString()
   {
        return size + " " + fly;
   }
}
```

return type

return method

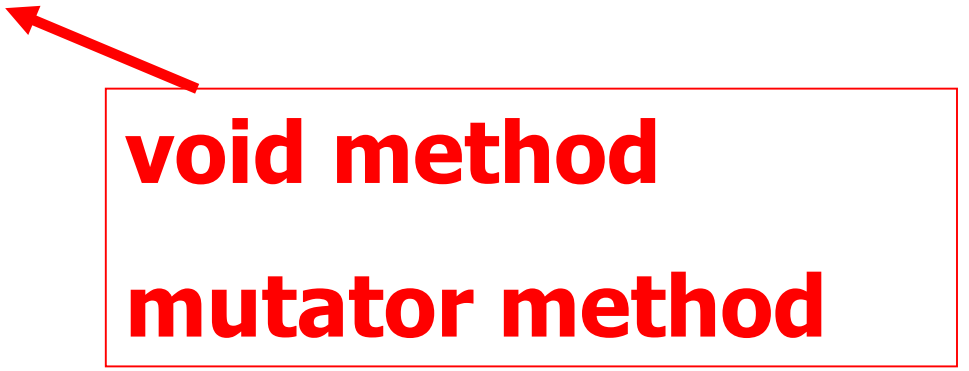
accessor method

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 changeSize( int s )
   {
        size = s;
   }
}
```



Mutator Methods

```
public void changeSize( int s )
{
    size = s;
}
```

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

turkey.java
turkeyrunner.java

Work on Programs!

Crank Some Code!

A+ Computer Science

WRITING CLASSES