Primitive DataTypes Example

```java
int x = 5;
System.out.println(x);
```
**Primitive DataType Example**

```java
int x = 5;
System.out.println(x);  //prints 5

int y = x;
System.out.println(y);
```
Primitive Data Type Example

```java
int x = 5;
System.out.println(x); // prints 5

int y = x;
System.out.println(y); // prints 5

y++;
System.out.println(y);
```
Primitive Data Type Example

```java
int x = 5;
System.out.println(x); // prints 5

int y = x;
System.out.println(y); // prints 5

y++; System.out.println(y); // prints 6
System.out.println(x);
```
Primitive Data Type Example

```java
int x = 5;
System.out.println(x);  // prints 5

int y = x;
System.out.println(y);  // prints 5

y++;
System.out.println(y);  // prints 6

System.out.println(x);  // prints 5
```
Primitive DataType Example

What’s happening?
Primitive DataType Example

What’s happening?

Each variable gets its own copy of the data:

```java
int x = 5;  //create variable called x, set
            //its value to 5.

int y = x;  //create variable called y, set
            //its value to whatever is
            //currently in x. They each have
            //their own copy of that value.
```
Person p1 = new Person(“Sally”);
System.out.println(p1.getName());
Person Object Example

Person p1 = new Person("Sally");
System.out.println(p1.getName());  //Sally

Person p2 = new Person("Joe");
System.out.println(p2.getName());
Person Object Example

Person p1 = new Person("Sally");
System.out.println(p1.getName());  //Sally

Person p2 = new Person("Joe");
System.out.println(p2.getName());  //Joe

p1 = p2;
System.out.println(p1.getName());
Person Object Example

Person p1 = new Person(“Sally”);
System.out.println(p1.getName());     //Sally

Person p2 = new Person(“Joe”);
System.out.println(p2.getName());     //Joe

p1 = p2;
System.out.println(p1.getName());     //Joe

p2.changeName(“Tom”);
System.out.println(p1.getName());
Person Object Example

```java
Person p1 = new Person("Sally");
System.out.println(p1.getName());  //Sally

Person p2 = new Person("Joe");
System.out.println(p2.getName());  //Joe

p1 = p2;
System.out.println(p1.getName());  //Joe

p2.changeName("Tom");
System.out.println(p1.getName());  //Tom
```
Person Object Example

What’s happening?

Each variable *still* gets its own copy of the data...but what is the data?

```java
Person p1 = new Person("Sally");
//p1 contains...
```
Person Object Example

What’s happening?

Each variable \textit{still} gets its own copy of the data...but what is the data?

```java
Person p1 = new Person("Sally");
//p1 contains an address like @Person4E32X
```
Person Object Example

What’s happening?

Each variable **still** gets its own copy of the data...but what is the data?

```java
Person p1 = new Person("Sally");
//p1 contains an address like @Person4E32X

Person p2 = p1;
```
Person Object Example

What’s happening?

Each variable *still* gets its own copy of the data...but what is the data?

```java
Person p1 = new Person("Sally");  
//p1 contains an address like @Person4E32X

Person p2 = p1;  
//p2 contains the same address that is currently in p1.  i.e. @Person4E32X
```
Person Object Example

What’s happening?

Each variable *still* gets its own copy of the data...but what is the data?

Person p1 = new Person("Sally");
Person p2 = p1;
p2.changeName("Tim");
//p2 contains an address of a Person object.
//We go to that address and call the
//changeName() method on that object. But,
//it is the same physical object that p1 is
//pointing to. Thus...
Person Object Example

What’s happening?

Each variable *still* gets its own copy of the data...but what is the data?

Person p1 = new Person("Sally");
Person p2 = p1;
p2.changeName("Tim");

System.out.println(p1.getName());
//prints Tim