Reference Variables
Problem 1:
Primitive vs. Reference Variables
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;
```
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;
```
### Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;
```
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;
```
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;

String s;
s = "Hello";
```
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;

String s;
s = "Hello";
```
Primitive vs. Reference Variables

```java
int x;
boolean b;
x = 17;
b = false;

String s;
s = "Hello";
```
Problem 2:
Reference Variables vs. Objects
Reference Variables

What happens in memory when the following code is executed given the definition of the `Foo` class on the side.

```java
public class Foo {
    private int n;
    public Foo ( int n ) {
        this.n = n;
    }
    public void set( int n) {
        this.n = n;
    }
    public int get( ) {
        return n;
    }
}

Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );
```
Reference Variables

What happens in memory when the following code is executed given the definition of the `Foo` class on the side.

```java
public class Foo {
    private int n;
    public Foo ( int n ) {
        this.n = n;
    }
    public void set( int n) {
        this.n = n;
    }
    public int get( ) {
        return n;
    }
}
```

```java
Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );
```
Reference Variables

What happens in memory when the following code is executed given the definition of the `Foo` class on the side.

```java
public class Foo {
    private int n;
    public Foo ( int n ) {
        this.n = n;
    }
    public void set( int n) {
        this.n = n;
    }
    public int get( ) {
        return n;
    }
}

Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );
```
Reference Variables

What happens in memory when the following code is executed given the definition of the Foo class on the side.

```java
public class Foo {
    private int n;
    public Foo ( int n ) {
        this.n = n;
    }
    public void set( int n ) {
        this.n = n;
    }
    public int get( ) {
        return n;
    }
}
```

Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );
Reference Variables

What happens in memory when the following code is executed given the definition of the `Foo` class on the side.

```java
public class Foo {
    private int n;

    public Foo ( int n ) {
        this.n = n;
    }

    public void set( int n) {
        this.n = n;
    }

    public int get( ) {
        return n;
    }
}

Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );
```
Reference Variables

What happens in memory when the following code is executed given the definition of the Foo class on the side.

```java
public class Foo {
    private int n;
    public Foo ( int n ) {
        this.n = n;
    }
    public void set( int n) {
        this.n = n;
    }
    public int get( ) {
        return n;
    }
}
```

Foo f1;
Foo f2;
f1 = new Foo (5);
f2 = f1;
f1.set(17);
System.out.println( f1.get() );
System.out.println( f2.get() );

Output: 17
17