Object-oriented Programming

- Objects: marry data with related methods
- Specifying a class does not create any objects
  - No memory allocation (or almost none)
- An object is an instance of a particular class
  - There can be many
- Objects have shared and independent parts
  - “static” indicates a shared part

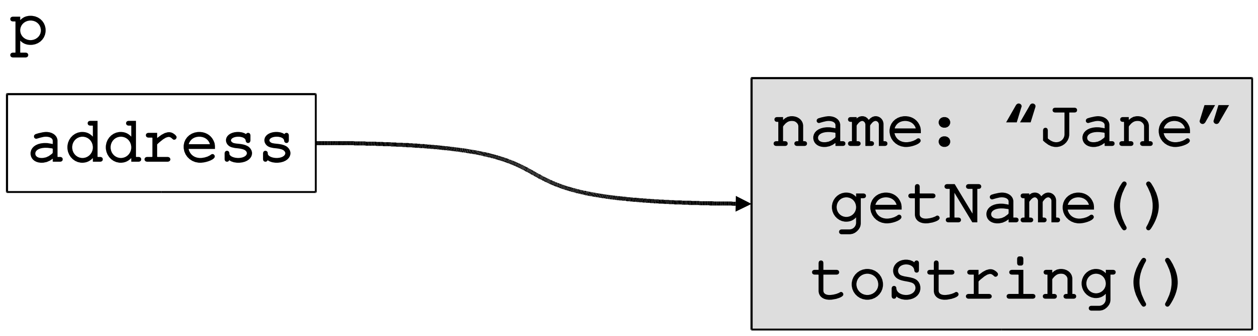
Class Person

```
public class Person {
    private String name="";          // no public variable
    public Person(String n) {         // "constructor"
        name=n;
    }
    public String getName() {          // Accessor method
        return name;
    }
    public String toString() {       // used to get a string representation
        return "name="+name;
    }
}
```

What does this class do??

Objects

- All Java objects extend Object (unless otherwise specified)
- An object is referred to by a reference variable
  - `Person p = new Person(“Jane”);`



Comparison and Use

```
public class PU {
   public static void main(String[] args) {
         Person p1 = new Person("Jane");
         Person p2 = new Person("Dick");
         Person p3 = new Person("Jane");
         Person p4 = p1;

         System.out.println("See " + p1 + " run. See " + p2 + " run");
         System.out.println(p1==p2);
         System.out.println(p1==p3);
         System.out.println(p1==p4);
         System.out.println(p1.getName()==p3.getName());
         System.out.println(p1.getName().equals(p3.getName()));
   }
}
```

Danger here!!!

A method of the class String

Output?

The `equals()` method

```
public boolean equals (Person p) {
  return name.equals(p.getName());
}

if (p1.equals(p2))
```

- Comparison of any objects should be done through implementing the `equals()` method

The `this` keyword

- Refers to the object through which the method is invoked

```
public class Person2 {
    private String name="";
    public Person2(String n) {name=n; }
    public String getName() {return name;}
    public String toString() {return "name="+name;}

    private Person2 spouse=null;
    public void marry(Person2 p) {
            spouse=p;
            p.marry(this);
    }
}
```

Example: marriage

```
public class PU2
{
   public static void main(String[] args)
   {
          Person2 p1 = new Person2("Jane");
          Person2 p2 = new Person2("Dick");
          p1.marry(p2);
   }
```

What does this do?

`null`

- `null` is a special word; it signifies no reference
- Must check for `null`, otherwise may cause `nullPointerException`

```
// Person3 is as person2 but with some changes/additions
public String toString() {
            if (spouse==null)
                return "name="+name+ ”   --single";
            else
                return "name="+name + " --married to " + spouse.getName();
   }
   private Person3 spouse=null;
   public void marry(Person3 p) {
            if (spouse != null) {
                spouse=p;
                p.marry(this);
            } }
```

finally: divorce

Create a new class Person4 that is the same a Person3 but has a divorce method

```
public class Person4 {
....
public void divorce(                                        ) {
            if (spouse != null) {
                spouse.divorce();
                spouse=null;
            }
  }
```

The keyword `static`

- Non-static vars/methods bind to the current object,
- `static` methods or variables belong to the entire class, and are shared
- Calls to `static` methods outside of class must prefix with classname
- Static methods and variables are almost always public
- Cannot use non-static methods and variables from a static method
- Almost always should only use static variables as constants

Static Variables -- example

```
public class St
{
   public static int si;
   private int ci;
   public St(int i)
   {
         ci=i;
   }
   public String toString()
   {
         return si+"<>"+ci;
   }
}
```

```
public class StU
{
   public static void main(String[] args)
   {
         St s1 = new St(5);
         St s2 = new St(6);
         System.out.println(s1+" " + s2);
         s1.si=7;
         System.out.println(s1+" " + s2);
         s2.si=8;
         System.out.println(s1+" " + s2);
         St.si=9;
         System.out.println(s1+" " + s2);
   }
}
```

Output of St, Stu?