Defining a simple Class

Making a Better SSN Class

- Create a separate class for an “SSN” object
- An object is defined by its attributes and behavior
- “Attributes” are data values
- “Behavior” is defined by the methods

Attributes (data values)

- The SSN value (String)

Behavior (methods)

- Constructor
- Set and Get methods

```
public class SSN {

   private String SSNumber;

public SSN (String s) {

   SSNumber = s;

}

public void setSSN(String s){

   SSNumber = s;

}

public String getSSN(){

   return SSNumber;

}
```

Error Checking

- The reason the data value is private is to ensure that the object contains correct data values. Public data values can be changed by the user.
- Methods that assign values to the data values should check for validity and throw an exception if they are not valid.

```
public SSN (String s) {
  if (!isValidSSN(s))
    throw new IllegalArgumentException("Invalid SSN.");
  else
    SSNumber = s;
}
```

```
public void setSSN(String s){
if (!isValidSSN(s))
    throw new IllegalArgumentException("Invalid SSN.");
  else
    SSNumber = s;
}
```

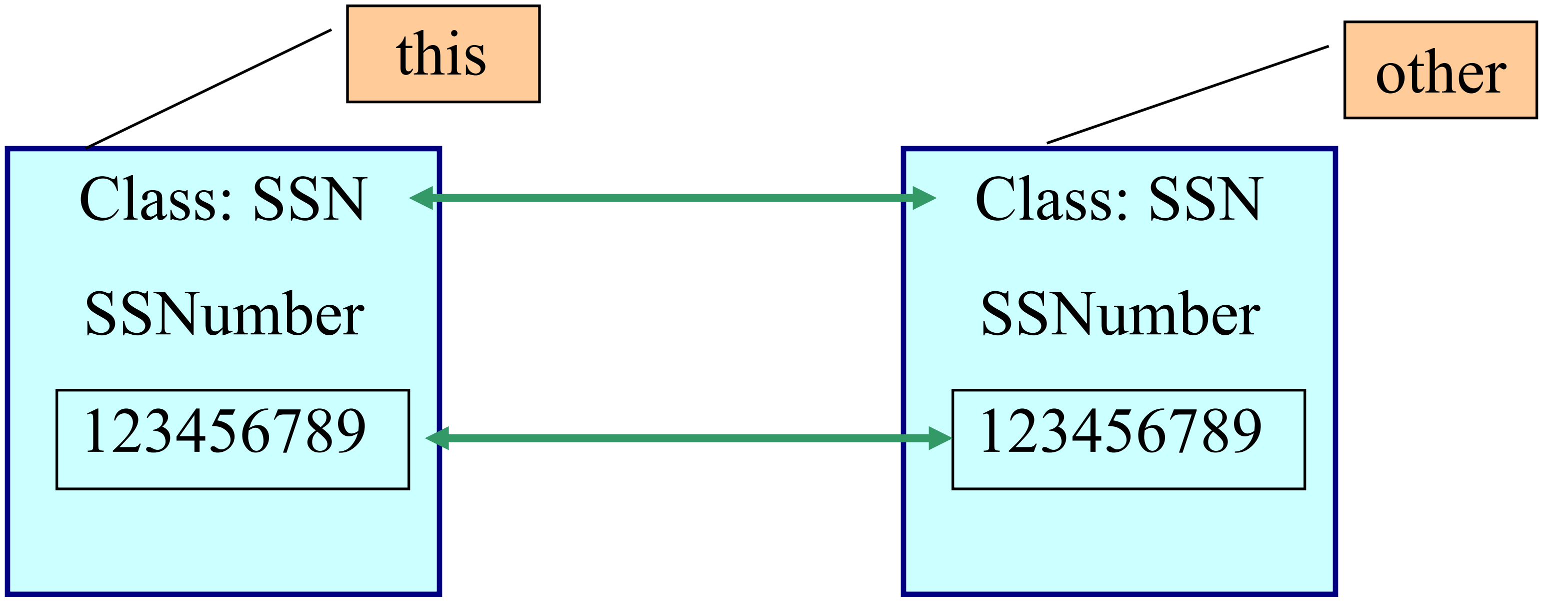
```
private static boolean isValidSSN(String s) {
      if (s.length() != 9)
         return (false);

      for (int i=0;i<9;i++)
         if (! Character.isDigit(s.charAt(i)))
            return(false);

      return (true);
} // isValidSSN
```

- The method is *private* because it is not to be called from outside.
- The method is *static* because it is not the behavior of an object

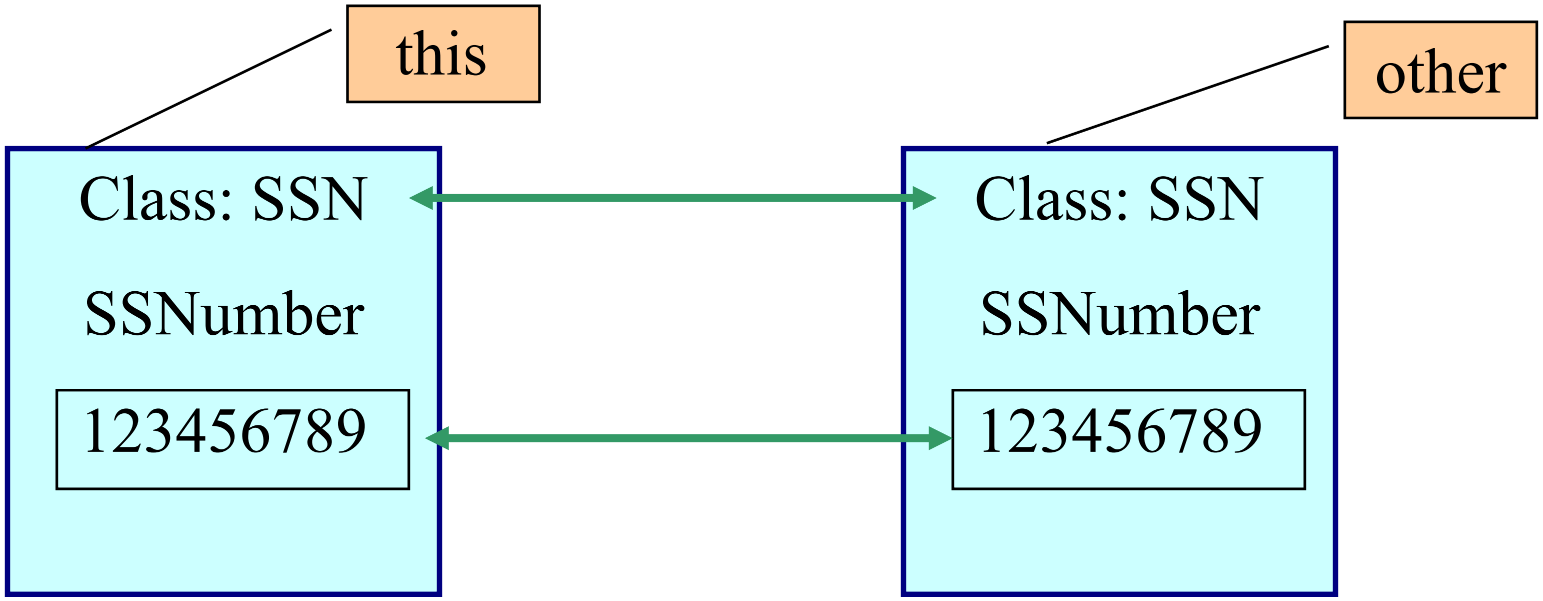
Overriding Methods of Class Object

- Since all classes inherit from class *Object*, the SSN class automatically has methods:
  - equals(Object o)
  - toString()
- These methods may not behave the way we want them to.

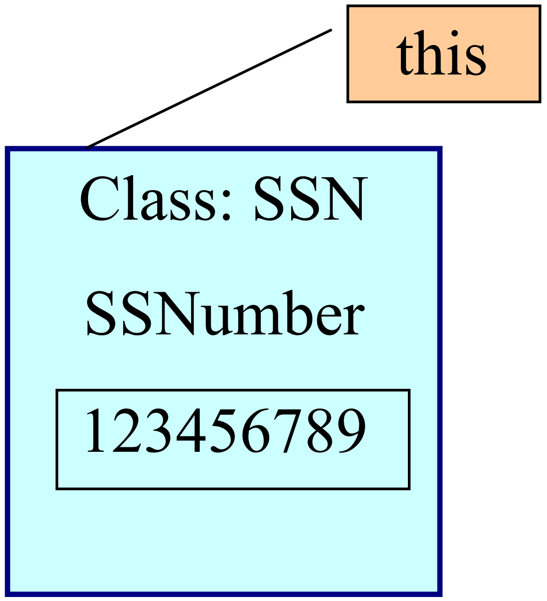
```
public boolean equals(Object other) {
  return (   other != null
          && getClass() == other.getClass()
          && SSNumber.equals(((SSN) other).SSNumber)
          );
 }
```



```
public int compareTo(SSN other) {
      return SSNumber.compareTo(other.toString());
  }
```



```
public String toString() {
      return SSNumber;
  }
```



The `this` operator

```
public SSN (String SSNumber) {
  if (!isValidSSN(s))
    throw new IllegalArgumentException("Invalid SSN.");
  else
    SSNumber = SSNumber;
}
```

Instance variable?

Formal parameter?

Formal parameter?

Instance variable?

The `this` operator is a reference to the class in which it is used.

```
public SSN (String SSNumber) {
  if (!isValidSSN(s))
    throw new IllegalArgumentException("Invalid SSN.");
  else
    this.SSNumber = SSNumber;
}
```

Instance variable!

Formal parameter!