Lecture 11

We write yet another class definition: the `Borrower` class. The way that objects created from the `Borrower` class definition send messages to instances of the `LibraryBook` class is interesting.

We consider how we can give each `Borrower` object a unique identification number. This leads us into a discussion of *class variables*.

What do we want the `Borrower` class to offer?

Some client code

```
LibraryBook b = new LibraryBook(
   new PersonName("Sir", "Hugh", "Jeegoh"),
   "Java through Dance", 3);

Borrower brrwr1 = new Borrower(new
   PersonName("Mr.", "Duncan", "Disorderly")

brrwr1.display();
brrwr1.takeOut(b);
brrwr1.display();
brrwr1.bringBack();

Borrower brrwr2 = new Borrower(
   new PersonName("Ms.", "Polly", "Morphism"
Borrower brrwr3 = new Borrower(
   new PersonName("Dr.", "Ben", "Dover"));
brrwr2.display();
brrwr3.display();
```


The `Borrower` interface

Constructor

```
public Borrower(PersonName aName)
```

Getter

```
public void display()
```

Setters

```
public void takeOut(LibraryBook aBook)
public void bringBack()
```

(A useful class might offer many more.)

The `Borrower` class definition

(Comments essential but omitted to save space)

```
public class Borrower
{
/* ====================================
       CONSTRUCTORS
   ====================================
*/
   public Borrower(PersonName aName)
   {  name = aName;
   }

/* ====================================
       PUBLIC INTERFACE
   ====================================
*/
   public void display()
   {


   }
```



```
   public void takeOut(LibraryBook aBook)
   {
   }

   public void bringBack()
   {
   }

/* ====================================
       INSTANCE VARIABLES
   ====================================
*/
   /**
    * The borrower's name.
    */
   private PersonName name;

   /**
    * The book the borrower has on loan
    * (null if s/he has no book on loan)
    */
   private LibraryBook bookOnLoan;
}
```


Unique identifiers for `Borrower` objects

- Suppose each borrower must have a unique number, starting from 1

- `Borrower` needs another instance variable

```
/**
 * The borrower's unique number.
 */
private int borrowerNum;
```

- We need another variable to keep track of what number to use for the next borrower. Then, the constructor will say:

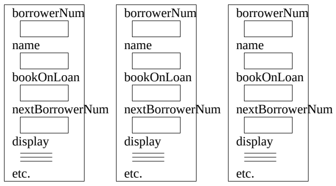
```
borrowerNum = nextBorrowerNum;
nextBorrowerNum++;
```

`nextBorrowerNum` as an instance variable

- If this variable is an instance variable:

```
/**
 * The number we will use as the borrower
 * number for the next borrower we create
 */
private int nextBorrowerNum = 1;
```

- Create `brrwr1`, `brrrwr2` and `brrwr3`:

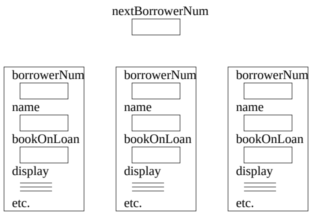


`nextBorrowerNum` as a class variable

- If this variable is a class variable:

```
/**
 * The number we will use as the borrower
 * number for the next borrower we create
 */
private static int nextBorrowerNum = 1;
```

- Create `brrwr1`, `brrrwr2` and `brrwr3`:



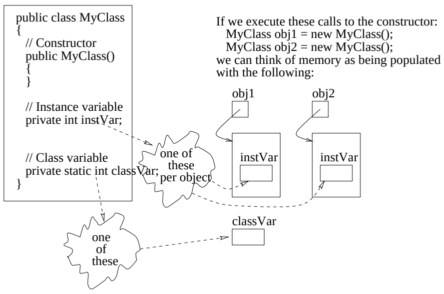
Summary of changes made to `Borrower`

```
public class Borrower
{
/* ====================================
      CONSTRUCTORS
   ====================================
*/ public Borrower(PersonName aName)
   {  name = aName;


   }

/* ========================================
      INSTANCE VARIABLES AND CLASS VARIABLE
   ========================================
*/

   private PersonName name;

   private LibraryBook bookOnLoan;

}
```


Class variables

- *Instance variables*: created per-instance (i.e. one in each object we create)
- *Class variables* (sometimes called 'static variables'): created per-class (i.e. the variable is 'shared' among all objects of the class)
- Class variables are very rare (except when `final`, see later)
- Declaration: at 'top-level' within a class definition, including the word `static`
- Initialisation: Java automatically initialises with a default value. But you can initialise them yourself in the declaration. (You can also use 'static initialisers'.)

Instance variables versus class variables



Class variables: scope of identifier, lifetime & visibility

- Scope: suppose the variable is called `nextBorrowerNum`
  - everywhere in this class definition, you can use `nextBorrowerNum`
  - in other class definitions, you should prefix with the class name, e.g. `Borrower.nextBorrowerNum` (*but you might still not be able to access the variable, depending on its visibility*)
- Lifetime (simplified): the whole of program execution
- Visibility: as per instance variables (`private` is preferred unless the variable is also `final`, in which case, it *might* be `public`)