Lecture 15

We write a simple class definition for the class of Ticket objects.

As we organise the computation (procedural abstraction), we settle upon a number of helper methods. But some of these are not *instance methods* (the kind we've been writing so far). Some are *class methods*.

We discuss class methods in detail and consider when we would use them.

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

Some client code

```
Ticket paganTicket = new Ticket(
   "Lughnasa",
   "1st August - Banshees welcome");
paganTicket.display();
System.out.println();
Ticket heathenTicket = new Ticket(
   "Imbolg (Free Chicken Sacrifices!)",
   "1st Feb");
heathenTicket.display(3);
```

The interface

```
public Ticket(String anOccasion, String aTim
public void display()
public void display(int aNum)
```


The `Ticket` class definition

(Comments essential but omitted to save space)

```
public class Ticket
{
/* ====================================
      CONSTRUCTORS
   ====================================
*/
   public Ticket(String anOccasion,
      String aTime)
   {



   }
```



```
/* ====================================
      PUBLIC INTERFACE
   ====================================
*/

/* ---Getters----------------------- */

   public void display()
   {



   }
```



```
   public void display(int aNum)
   {  for (int i = 0; i < aNum; i++)
      {  display();
         System.out.println();
      }
   }

/* ====================================
      HELPER METHODS
   ====================================
*/
   private void drawTextLine(String theText)
   {  System.out.println(VERT_BORDER +
        padString(theText, ' ', textWidth) +
        VERT_BORDER);
   }

   private static void drawHorizLine(
      int aNumOfChars, char aChar)
   {  for (int i = 0; i < aNumOfChars; i++)
      {  System.out.print(aChar);
      }
      System.out.println();
   }
```



```
   private static String padString(
      String str, char aChar, int reqdLength
   {  String paddedStr = "";
      int amntOfPadding =
         reqdLength - str.length();
      int leftPad = amntOfPadding / 2;
      int rightPad = (int)
         (Math.round(amntOfPadding / 2.0));
      for (int i = 0; i < leftPad; i++)
      {  paddedStr = paddedStr + aChar;
      }
      paddedStr = paddedStr + str;
      for (int i = 0; i < rightPad; i++)
      {  paddedStr = paddedStr + aChar;
      }
      return paddedStr;
   }
```

```
/* ========================================
      INSTANCE VARIABLES  & CLASS VARIABLES
   ========================================
*/
   // A description of the event.
   private String occasion;

   // A description of the start/end date/ti
   private String time;

   // The maximum number of characters wide
   // this ticket is (excluding borders).
   private int textWidth;

   // The horizontal border of a ticket.
   private static final char
      HORIZ_BORDER = '-';

   // The vertical border of a ticket.
   private static final char
      VERT_BORDER = '|';
}
```


Class methods

- *Instance methods*: created per-instance (i.e. one in each object we create) (At least, we think of things this way!)
- *Class methods* (sometimes called 'static methods'): created per-class (i.e. the method is 'shared' among all objects of the class)
- When defining class methods, include the keyword `static`

Invoking class methods

- Sending a message to invoke a class method called, e.g., `drawHorizLine`
  - everywhere in this class definition, just use, e.g.

    `drawHorizLine(12, '*');`
  - in other class definitions, you should prefix with the class name, e.g.

    `Ticket.drawHorizLine(12, '*');`

    (*but you might still not be able to invoke the method, depending on its visibility*)

Class methods: `public` or `private`?

- `public` or `private`? Will clients find the method useful or not?
- E.g. `public` class methods in the Java library (in `java.lang.Math`):

```
public static int max(int a, int b)
public static double random()
```

- As we've seen, clients invoke them by prefixing with class name:

```
textWidth = Math.max(
   occasion.length(), time.length());

return (int) (Math.random() * 6 + 1);
```


When do we use class/instance methods?
A rule of thumb

- If you are organising the computation of a *class method* into smaller methods,
  - the smaller methods will also be *class methods.*
- If you are organising the computation of an *instance method* into smaller methods,
  - the smaller methods *can* be *instance methods*
  - this *will* compile & run
  - but it misses the fact that some of these smaller methods could actually be class methods (which run more efficiently)

A more accurate rule

- If a method
  - does not invoke any instance methods in this class, and
  - does not access any instance variables in this class,

  then it can be a class method. (Otherwise, it's an instance method.)
- Alternative way of saying the same thing: if a method does not use `this` (either explicitly or even implicitly), then it can be a class method. (Otherwise, it's an instance method.)