CSE 142

Static Data and Methods

A Programming Task

- **Suppose we wish to give each BankAccount a unique serial number.**

```
class BankAccount {
    private String accountName;        // account holder's name
    private double balance;            // account balance
    private int accountNumber;         // unique account number
    ...
}
```


BankAccount Constructor (First Attempt)

- **We want the constructor to assign a unique account number to each newly created account.**

```
class BankAccount {
    ...
    // construct new BankAccount with given name, balance, and unique acct #
    public BankAccount(String accountName, double initialBalance) {
      this.accountName = accountName;
      this.balance = initialBalance;
      this.accountNumber = nextAvailableAccountNumber;
      nextAvailableAccountNumber ++;
    }
    ...
}
```

- **Questions: Where (what) is nextAvailableAccountNumber? Where is it stored?**

What is nextAvailableAccountNumber?

- **Instance variable?**
  - **No – we don't want one of these per object (class instance)**
- **Local variable in the constructor?**
  - **No – we need to retain next available value between creation of one object and the next**
- **Answer: we need a *single* copy somewhere, not associated with any particular object.**
- **Solution: have one copy that's associated with class BankAccount itself, not with individual instances.**

Static Variables

- **A *static* variable is one that belongs to the class itself, not to instances. It is shared by all instances.**

```
class BankAccount {
    // object instance variables:
    private String accountName;        // account holder's name
    private double balance;            // account balance
    private int accountNumber;         // unique account number

    // class variables:
    static private int nextAvailableAccountNumber = 1;  // next available acct #
    ...
}
```

- **Initialize the static variable in its declaration.**

BankAccount Constructor (Final Version)

- **Now we're all set:**

```
class BankAccount {
    ...
    // construct new BankAccount with given name, balance, and unique acct #
    public BankAccount(String accountName, double initialBalance) {
      this.accountName = accountName;
      this.balance = initialBalance;
      this.accountNumber = BankAccount.nextAvailableAccountNumber;
      BankAccount.nextAvailableAccountNumber ++;
    }
    ...
}
```

- **Can refer to a static variable without using class name. (Why?)**
- **Can even refer to it as this.nextAvailableAccountNumber (questionable style).**

Draw the Picture

```
BankAccount mine = new BankAccount("Teacher", 170.42);
BankAccount yours = new BankAccount("Former Student", 435769.17);
```


Symbolic Constants: Static Final Variables

- Sometimes we just want to give a name to a constant value, like pi or e or the number of cms per inch.
- Solution: a static variable, but further qualified with *final* so it can't be changed after it is initialized.

```
/** An important number */
public static final double PI = 3.1415926535;
```

  - Final variables must be initialized when declared; cannot be changed later.
  - Any variable that isn't changed after declaration can be marked final.

Constants in the Java Libraries

- Several Java classes contain useful named constants.
- Class Math contains PI and E, with the expected values.

```
this.area = Math.PI * this.radius * this.radius;
```

- Classes like Integer and Double contain things like the largest possible int value, the smallest positive non-zero double, etc.
- The Color class has static final variables for many predefined colors (Color.green, etc.).

Static Methods

- Some methods in Java aren't naturally associated with particular objects. (At least not reference objects.)
  - Basic math functions – sqrt, sin, cos, tan
- Other methods we might want to call before we've created any instance of a class, or that provide a way to create an object aside from a constructor.
  - newInputFromFile(String fileName) in the Input class
  - test methods
- Such methods can be declared *static*: the method is not part of any instance, but rather the class itself.
  - Invoked by sending a message to the class itself.
  - Cannot access `this` or any instance variables or methods inside a static method.

Class Math

- Example: Math (class in standard Java library)

```
public class Math {
    public static final double PI = 3.1415926535;
    public static final double E = 2.71828;
    public static double sqrt(double x) { ... }
    public static double sin(double x)  { ... }
    ...
}
```

- Example of use:

```
double distance = Math.sqrt(dx*dx + dy*dy);
```


Method main

- BlueJ and Jeva allow us to manipulate objects directly
- Standard Java applications: need to identify class that where execution is to start & it starts in method main of that class
- main must always be defined like this

```
public static void main(String[] args) { ... }
```

  - Typical contents of main: create some objects and call a method or two to get things going
  - args array contains any string arguments passed to the program when it was started. Actual name need not be args.