CSE 142

Static Variables 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
    …
}
```

- We'd like the constructor to assign unique account numbers as BankAccount objects are created
  - Don't want to depend on client code supplying the number

BankAccount Constructor

- First Attempt

```
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
  - Single, unique copy shared by all instances
  - Usually initialized in its declaration

```
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 #
    …
}
```

BankAccount Constructor (Fixed 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?)
- Java even allows this.nextAvailableAccountNumber (but that seems very misleading)

Draw the Picture

```
BankAccount mine = new BankAccount("Teacher", 170.42);
BankAccount yours = new BankAccount("Former Student", 4351769.17);
```

1/10/2003 (c) 2001-3, University of Washington S-7

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 gallons per liter
- 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 won't be changed after initialization can be marked final

1/10/2003 (c) 2001-3, University of Washington S-8

Constants in the Java Libraries

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

```
area = Math.PI * radius * 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.)

1/10/2003 (c) 2001-3, University of Washington S-9

Static Methods

- Some methods in Java aren't naturally associated with particular 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 since there's no associated object (instance)

1/10/2003 (c) 2001-3, University of Washington S-10

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);
```

1/10/2003 (c) 2001-3, University of Washington S-11

Method main

- We now can understand the definition of main methods

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

- Static – it belongs to a class, not an instance of a class
  - So it can be executed without creating any objects first
- 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 (command line or other interface, depending on particular implementation)
  - Actual name can be whatever you want, not necessarily args

1/10/2003 (c) 2001-3, University of Washington S-12