Odds and ends

Some features of Java we haven’t discussed

Static methods

// Example:
// Java’s built-in Math class
public class Math{
    public static int abs(int a) {
        if (a >= 0) {
            return a;
        } else {
            return -a;
        }
    }
    public static double toDegrees(double radians) {
        return radians * 180 / PI;
    }
}

// Using the class:
System.out.println(Math.abs(-5));
// didn’t need to create any object

Static methods

- **static**: Part of a class, not part of an object.
- **Static methods**:
  - Do not require an instance of the class and do not understand the *implicit parameter, this*; therefore, cannot access an object’s instance variables
  - good for code related to a class but not to each object’s state
  - if public, can be called from inside or outside the class

Static revisited
Static variables

- **static**: Part of a class, rather than part of an object.
- Classes can have static variables.
- Static variables are not replicated in each object; a single variable is shared by all objects of that class.

```java
private static type name;
```

or,

```java
private static type name = value;
```

- Example:
  ```java
  private static int count = 0;
  ```

Examples in the Java library

- Static variables in the **System** class:
  - System.in and System.out.
  (System is a class, and out is a static variable in that class, that has a method called println)
  
- And in the Java Math class:
  ```java
  public class Math {
  public static final double PI = 3.141592653589793;
  public static final double E = 2.718281828459045;
  ...
  }
  ```
  https://docs.oracle.com/javase/7/docs/api/java/lang/System.html

Example

- You are writing a class to represent a bank account, and you would like the constructor to automatically assign a running number as the account number.

- How can static variables help you?

Assigning ids for BankAccount

```java
public class BankAccount {
  // static variable for assigning an account number
  // (shared among all instances of the class)
  private static int lastAssignedNumber = 1000;
  // instance variables (replicated for each object)
  private float balance;
  private int id;

  public BankAccount(float initial_balance) {
    lastAssignedNumber++; // advance the id
    id = lastAssignedNumber; // give number to account
    balance = initial_balance;
  }
  ...
  public int getID() { // return this account's id
    return id;
  }
}
Static variables

- Exception: Static constants, which may be either private or public:
  - public class BankAccount
    - public static final double OVERDRAFT_FEE = 5;
    - // Refer to it as BankAccount.OVERDRAFT_FEE
  - Minimize the use of static variables (static final variables are ok)

Java features we haven’t discussed

- Packages
  - A package is a named collection of related classes that are grouped in a directory
  - Using code from a package:
    - import java.awt.Rectangle;
      - Rectangle rectangle = new Rectangle();

http://docs.oracle.com/javase/tutorial/java/package/index.html
Java features we haven’t discussed

- JUnit: a framework that lets you write tests for each method, then easily run those tests (unit testing)
- Martin Fowler: “Never in the field of software development was so much owed by so many to so few lines of code.”
- The standard tool for test-driven development in Java
- JUnit integration in Eclipse

Java features we haven’t discussed

- Annotations.
  - Provide information about a program
  - Example:
    ```java
    @Override
    public boolean equals(Object obj) {...}
    ```

If a method marked with @Override doesn’t override a method in one of its superclasses, the compiler generates an error.

Java features we haven’t discussed

- Final methods and classes
  - A final method cannot be overridden
  - A final class cannot be extended
    - Example: public final class String

Java features we haven’t discussed

- Generics.
  - You’ve had a taste – more in CS200.
Exceptions revisited

- Until now you only used predefined Java exceptions.
- You can write your own!
- Why would you want to do that?

Example

```java
public class DivideByZeroException extends Exception {
    public DivideByZeroException() {
        super("Divide by zero");
    }
    public DivideByZeroException(String message) {
        super(message);
    }
}
```