Announcements

- A1 is due tomorrow
  If you are working with a partner: form a group on CMS & submit once

- A2 is out. Remember to get started early!
  Because of Jewish holidays, the due date for A2 has been changed to Sunday, 16 September.
  ONLY TWO DAYS OF LATENESS ALLOWED.
  Last day to submit is 18 September.

- Next week's recitation is on testing. No tutorial/quiz this week!
Example Constructor: Person Class

```java
public class Person {
    private String firstName; // cannot be null
    private String lastName;

    /** Create a person with the given names. */
    public Person(String f, String l) {
        assert f != null;
        firstName = f;
        lastName = l;
    }
}
```

**Constructor:**
Initializes fields to make class invariants true

**JavaHyperText**
Filter: constructor
HERE is a summary of all important points concerning constructors: pdf file.
Adding a Middle Name Option (v1)

```java
public class Person {
    private String firstName; // cannot be null
    private String middleName;
    private String lastName;

    public Person(String f, String l) {
        assert f != null;
        firstName = f;
        lastName = l;
    }

    public Person(String f, String m, String l) {
        assert f != null;
        firstName = f;
        middleName = m;
        lastName = l;
    }
}
```

Want to change body to call first constructor
public class Person {
    private String firstName; // cannot be null
    private String middleName;
    private String lastName;

    public Person(String f, String l) {
        assert f != null;
        firstName = f;
        lastName = l;
    }

    public Person(String f, String m, String l) {
        this(f, l);
        middleName = m;
    }
}

Use this (not Person) to call another constructor in the class.
Must be first statement in constructor body!
public class Person {
    private String firstName; // cannot be null
    private String lastName;

    public Person() {}; // default constructor

    public String toString() {
        return firstName + " " + lastName;
    }

    Person p = new Person();
}
Wow, every class has an empty Constructor?

```java
public class Person {
    private String firstName; //not null
    private String middleName;
    private String lastName;
    public Person(String f, String l) {
        assert f != null;
        firstName = f;
        lastName = l;
    }
    public Person(String f, String m, String l) {
        this(f, l);
        middleName = m;
    }
}\n```

Person p = new Person();

Nope!

Syntax Error: No constructor in Person matches this invocation
Arguments: ()
Expected return type: Person
Candidate signatures:
Person(String, String)
Person(String, String, String)
public class Person {
    private String firstName;
    private String lastName;

    public Person(String f, String l) {
        assert f != null;
        firstName = f; lastName = l;
    }

    public String toString() {
        return firstName + " " + lastName;
    }
}

Person p1 = new Person("Grace", "Hopper");
p1.toString();
public class Cornellian extends Person {
    private String netID;

    /** Constructor: Person with a netID. */
    public Cornellian(String f, String l, String id) {
        super(f, l);
        netID = id;
    }
}

new Cornellian("David", "Gries", "djg17");
public class Cornellian extends Person {
    private String netID;

    /** Constructor: Person with a netID. */
    public Cornellian(String f, String l, String id) {
        super();
        netID = id;
    }
}

new Cornellian("David", "Gries", "djg17");
Within a subclass object, `super` refers to the partition above the one that contains `super`.

Because of keyword `super`, the call `toString` here refers to the `Person` partition.

```
class Person {
    string netID; // Student ID
    Person(string netID) : netID(netID) {} // Constructor

    string toString() { return super.toString() + " " + netID; }
}
```
Without OO ...

Without OO, you would write a long involved method:

```java
public double getName(Person p) {
    if (p is a CornellUndergrad)
        { ... }
    else if (p is a CornellFaculty)
        { ... }
    else if (p prefers Cornellian)
        { ... }
    else ...
}
```

OO eliminates need for many of these long, convoluted methods, which are hard to maintain. Instead, each subclass has its own `getName`. Results in many overriding method implementations, each of which is usually very short.
/** Return middle value of a, b, c (no ordering assumed) */

public static int middle(int a, int b, int c) {
    if (b > c) {
        int temp = b;
        b = c;
        c = temp;
    }

    if (a <= b) {
        return b;
    }

    return Math.min(a, c);
}

Parameter: variable declared in () of method header

Local variable: variable declared in method body

a 8 b 6 c 7

All parameters and local variables are created when a call is executed, *before* the method body is executed. They are destroyed when method body terminates.
/** Return middle value of a, b, c (no ordering assumed) */

public static int middle(int a, int b, int c) {
    if (b > c) {
        int temp = b;
        b = c;
        c = temp;
    }
    if (a <= b) {
        return b;
    }
    return Math.min(a, c);
}

Scope of local variable (where it can be used): from its declaration to the end of the block in which it is declared.
**Scope In General: Inside-out rule**

*Inside-out rule:* Code in a construct can reference names declared in that construct, as well as names that appear in enclosing constructs. (If name is declared twice, the closer one prevails.)

```java
/** A useless class to illustrate scopes*/

public class C {
    private int field;
    public void method(int parameter) {
        if (field > parameter) {
            int temp = parameter;
        }
    }
}
```
/** Return middle value of a, b, c (no ordering assumed) */

public static int middle(int a, int b, int c) {
    int temp;
    if (b > c) {
        temp = b;
        b = c;
        c = temp;
    }
    if (a <= b) {
        return b;
    }
    return Math.min(a, c);
}
public class ScopeQuiz {
    private int a;
    public ScopeQuiz(int b) {
        System.out.println(a);
        int a = b + 1;
        this.a = a;
        System.out.println(a);
        a = a + 1;
    }
    public static void main(String[] args) {
        int a = 5;
        ScopeQuiz s = new ScopeQuiz(a);
        System.out.println(s.a);
    }
}