Announcements

- A1 is due Thursday
  If you are working with a partner: form a group **well before submitting** on CMS. After forming group, **only one** you needs to submit.
- This week's recitation is on testing. No tutorial/quiz this week.
- A2 is out now
Today’s Topics

- Scope, local variables, inside-out rule
- Overloading, bottom-up rule
- Constructors, this, default, super

All searchable in JHT!

JavaHyperText

Filter:
constructor

HERE is a summary of all important points concerning constructors: pdf file.
/** 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);
}
/** 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);
}
**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
public class C {
    private int field;
    public void method(int parameter) {
        if (field > parameter) {
            int temp = parameter;
        }
    }
}
```
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);
    }
}
Review: Constructor

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

    /** Constructor: a Person with first and last names f, l.
     * Precondition: f is not null. */
    public Person(String f, String l) {
        assert f != null;
        firstName = f;
        lastName = l;
    }
}
```

**Constructor:**
Initialize fields to truthify class invariant
Review: Which `toString` is called?

```java
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 `pl` = new Person("Grace", "Hopper");
pl.ToString();
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();
Grace Hopper

(1906-1992)
Rear Admiral, US Navy
PhD, Math, Yale, 1934
Pioneering computer programmer
Posthumous Presidential Medal of Freedom (highest civilian honor), 2016
Grace Hopper: “bug”
**Middle Names (v1)**

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

    /** Constructor: ... */
    public Person(String f, String l) {
        assert f != null; firstName= f; lastName= l; }

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

Better: reuse first constructor without copying code.
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;
    }
}

Use this (not Person) to call another constructor in the class.
Must be first statement in constructor body!
Default Constructor

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

    public Person() {};
}

Person p = new Person();
```
Default Constructor

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;
    }
}

Person p = new Person();
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;
    }

    // Method implementation...
}

new Cornellian("Bill", "Nye", "bn29");
public class Cornellian extends Person {

private String netID;

/** Constructor: Person with a netID. */
public Cornellian(String f, String l, String id) {
super();
netID = id;
}
}
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.
A ship in port is safe, but this is not what ships are built for. Sail out to sea and try new things.

-- Rear Admiral Grace Hopper