Today's Topics

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

All searchable in JHT!

Local Variables

```java
/** 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 Variables

```java
/** 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 In General: Inside-out rule

```java
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.)
```
What 3 numbers are printed?

```java
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);
    }
}
```

A: 5, 6, 6
B: 0, 6, 6
C: 6, 6, 6
D: 0, 6, 0

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

Person p1 = new Person("Grace", "Hopper");
p1.toString();

Which toString : Bottom-up Rule

Which method is used? Start at bottom of the object and search upward until you find a match.

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

Middle Names (v2)

```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;
    }
}
```

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

```java
Person p = new Person();
```

```
Syntax Error: No constructor in Person matches this invocation
Arguments: ()
Expected return type: Person
Candidate signatures:
Person(String, String)
Person(String, String, String)
```

```
Nope!
```

Super Constructor

```java
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;
    }
}
```

```java
new Cornellian("Bill", "Nye", "bn29");
```

Use super (not Person) to call superclass constructor.
Must be first statement in constructor body!

```
If first statement in constructor body is not a constructor call, Java inserts super(); for you!
```

Super Constructor

```java
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.

**Grace Hopper**

A ship in port is safe, but that is not what ships are built for.
Sail out to sea and try new things.

-- Rear Admiral Grace Hopper