Local vars; Inside-out rule; constructors

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```

Homework

1. Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

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);
}
```

Principle: declaration placement

```java
/** 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);
}
```

Homework

Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```

Homework

Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```

Homework

Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```

Homework

Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```

Homework

Visit course website, click on Resources and then on Code Style Guidelines. Study
   4.2 Keep methods short
   4.3 Use statement-comments …
   4.4 Use returns to simplify method structure
   4.6 Declare local variables close to first use …

References to text and JavaSummary.pptx

- Local variable: variable declared in a method body
  B.10–B.11 slide 45
- Inside-out rule, bottom-up/overriding rule C.15 slide 31–32
  and consequences thereof slide 45
- Use of this B.10 slide 23–24 and super C.15 slide 28, 33
- Constructors in a subclass C.9–C.10 slide 24–29
- First statement of a constructor body must be a call on another
  constructor — if not Java puts in super(); C.10 slide 29

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);
}
```
Assertions promote understanding

```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;  // swap temp and b
        b = c;
        c = temp;
    }
    // b <= c
    if (a <= b) {
        return b;
    }
    return Math.min(a, c);
}
```

**Assertion:** Asserting that \( b \leq c \) at this point. Helps reader understand code below.

Bottom-up/overriding rule

Which method `toString()` is called by `turing.toString()`?

**Overriding rule or bottom-up rule:** To find out which is used, start at the bottom of the object and search upward until a matching one is found.

Calling a constructor from a constructor

```java
public class Time
    private int hr; // hour of day, 0..23
    private int min; // minute of hour, 0..59
    /** Constructor: instance with h hours and m minutes */
    public Time(int h, int m) {
        hr = h; min = m; assert ...;
    }
    /** Constructor: instance with m minutes ... */
    public Time(int m) {
        this(m / 60, m % 60);
    }
...
```

Want to change body to call first constructor

Calling a constructor from a constructor

```java
public class Time
    private int hr; // hour of day, 0..23
    private int min; // minute of hour, 0..59
    /** Constructor: instance with h hours and m minutes */
    public Time(int h, int m) {
        hr = h; min = m; assert ...;
    }
    /** Constructor: instance with m minutes ... */
    public Time(int m) {
        this(m / 60, m % 60);
    }
    Use this (not Time) to call another constructor in the class.
    Must be first statement in constructor body!
```

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
Person a0
    name
    getNameWithSep() {
        return name + sep;
    }

Person a1
    name
    getNameWithSep() {
        return name + sep;
    }
```

Constructing with a Superclass

```java
/** Constructor: person "f n" */
public Person(String f, String l) {
    first= n;
    last= l;
    Use super (not Person) to call superclass constructor.
}
/** Constructor: PhD "Dr. f m, l" */
public PhD(String f, char m, String l) {
    super(f, l);
    middle= m;
    Must be first statement in constructor body!
}
new PhD("Ross", 'E', "Tate");```
About super

Within a subclass object, `super` refers to the partition above the one that contains `super`.

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

Without OO …

Without OO, you would write a long involved method:

```java
public double getName(Person p) {
    if (p is a PhD)
        { … }
    else if (p hates formality)
        { … }
    else if (p prefers anonymity)
        { … }
    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.