Last Time

• We developed a Clock class that included
  – Some instance variables, declared as private
  – A constructor that takes three parameters: hours, minutes, and seconds
  – A constructor that takes no parameters
  – A toString() method
  – Some class constants
• And we wrote a test program that created and manipulated instances of Clock

This Time

• We will look at
  – Passing objects as parameters to methods
  – Accessing another object’s private instance variables
  – Calling methods
  • The syntax and rules for doing so
• And we will use these concepts to modify Clock

Passing Objects to Methods

• As with Arrays, objects are references
  – So, when we pass an object to a method, we can change its value

class PassingReferences {
  public static void main(String[] args) {
    StringBuffer abuf = new StringBuffer("testing");
    System.out.println("abuf is now " + abuf);
    modify(abuf);
    System.out.println("abuf is now " + abuf);
  }
  static void modify(StringBuffer sb) {
    sb.append("1 2 3");
  }
}
Object state after `sb.append()`

Variables in `main()`  
Variables in `modify(sbuf)`

- `sbuf` → `testing, 1 2 3`  
  StringBuffer object

Object state before `new`

Variables in `main()`  
Variables in `modify(sbuf)`

- `sbuf` → `testing`  
  StringBuffer object

Object state after `new`

Variables in `main()`  
Variables in `modify(sbuf)`

- `sbuf` → `testing, 1 2 3`  
  StringBuffer object
- `sb` → `doesn't work`  
  StringBuffer object

Passing Objects to Methods

* But, you can't change the value of the reference

```java
class ModifyParameters {
    public static void main(String[] args) {
        StringBuffer sbuf = new StringBuffer("testing");
        System.out.println("sbuf is now " + sbuf);
        modify(sbuf);
        System.out.println("sbuf is now " + sbuf);
    }
    static void modify(StringBuffer sb) {
        sb = new StringBuffer("doesn't work");
    }
}
```
Accessing Another Object's Private Variables

• An instance method operating on one object can access another object's private fields
  – Remember, private limits variable access to methods in the class
  – Implementation details are hidden from other classes, but not from objects in the same class
  – Let's create an add() method in Clock

```
Clock when = new Clock( 15, 30, 0 );
Clock howLong = new Clock( 1, 0, 0 );
when.add( howLong );
```

Calling Methods

• There are three ways to call a method, depending on
  – whether or not the method is in the same class
  – whether the method is an instance method or a class method

Calling Methods in the Same Class

• If a method is in the same class as the calling method, just use the method name
  – a = foo( b );
  – Instance Instance
  – Instance Class
  – Class Class
  – Class cannot call Instance
  • Why not?

Calling Methods in Another Class

• From another class:
  – To call an instance method, you must use the name of an instance of the class
    ```java
    String s = "abc";
    int a = s.length();
    ```
  – To call a class method, you must use the name of the class
    ```java
    double r = Math.random();
    ```
Calling Methods

- Let's add a method tick() to our clock class
  - Advance the time 1 second
  - Call another method in Clock to implement this method.

Quiz 5: Friday March 7

- This quiz will cover sections 6.1 to 6.10 and 6.12 of the book
- You should be able to answer questions about
  - Standard classes such as String and StringBuffer
  - Elements of a class
  - Abstract Data Types
  - Instance variable
  - Class variables, and Class constants
  - Instance methods
  - Class methods

Quiz 5: Sample Question

- Create the class Point needed by the following program, so that the output is the single line: 3, 5

```java
class PointTest {
    public static void main(String[] args) {
        Point p = new Point(3, 5);
        System.out.println(p);
    }
}
```
Quiz 5: Sample Questions

• True or False: public methods can only access public instance variables.

• True or False: Java always creates a default, no-argument constructor for you when you define a class.

• True or False: An accessor method is one that returns a value from an object without changing the object's value.

Quiz 5: Sample Questions

• What is printed by the following program?

class Quiz5Example1 {
    public static void main( String[] args ) {
        String s = "abcdefg";
        StringBuffer sbuf = new StringBuffer("abc");
        System.out.println( s.charAt( 3 ) );
        System.out.println( s.length() );
        System.out.println( s.indexOf("efg") );
        System.out.println( s.indexOf('z') );
        sbuf.append('z');
        System.out.println( sbuf.length() );
        sbuf.setCharAt( 2, 'x' );
        System.out.println( sbuf );
    }
}

Quiz 5: Sample Question

• What is printed by the following program?

class Quiz5Example2 {
    public static void main( String[] args ) {
        Point p1 = new Point( 2, 3 );
        Point p2 = new Point( 5, 6 );
        p1.mystery( p2 );
        System.out.println( p1 );
        System.out.println( p2 );
        p2.whoKnows( p1 );
        System.out.println( p1 );
        System.out.println( p2 );
    }
}

class Point {
    private int x;
    private int y;
    public Point( int x1, int y1 ) { x = x1; y = y1; }
    public void mystery( Point p ) { x = x + p.x; y = y + p.y; }
    public void whoKnows( Point p ) { x = p.x + 3; y = p.y + 2; }
    public String toString() { return x +", " + y; }
}
Quiz 5: Sample Question

• Given the following class
  
  ```java
  class Counter {
    private static int count = 0;
    public static int getCount() {
      return count;
    }
  }
  ```

• How do you access `count` from outside the class?

A) int c;
   Counter ctr = new Counter();
   c = ctr.getCount();

B) int c = Counter.getCount();

C) int c = Counter.count;

D) int c = count;

---

Quiz 5: Sample Question

• Given the following class
  
  ```java
  class Counter {
    private int count = 0;
    public Counter(int c) { count = c % 100; }
    public static int getCount() {
      return count;
    }
  }
  ```

• Add a class constant `MAXCOUNTER` with a value of 100, and modify the class to use the constant.