CS180 Recitation

Defining Your Own Classes II
Announcements

• Project 5 is out
  – 2 weeks to finish.
  – Milestone due on Wednesday.

• Exam
  – Handed back today.

• Class next week
  – Prof. Prabhakar is traveling.
  – Class will be held at the same time.
  – Lectures will be given by Barry Whitman on Monday and Prof. Clifton on Wednesday.
Exam Postview

• Which of the following is NOT a 'type' in Java?
  – void.
  – int.
  – Integer.
  – Static. * (keyword, not a type)
What is the output of the following code if the input string is “CS 180”.
Scanner scanner = new Scanner();
String str = scanner.next();
System.out.print(str);

CS180.
CS180.
CS.
CS 180.
The above code fragment does not compile.* Scanner needs argument.
Exam Postview

• What is the value of y when this code is executed?

```java
int x = 4;
int y = Math.ceil(x % 5 + x / 5.0);
```

– 1.
– 6.
– 5. * precedence rules work left to right
– 4.
int n=20; int p = n+5; int q = p-10;
int r = 2 * (p-q);
switch(n) {
    case p: n = n+1;
    case q: n = n+2;
    case r: n = n+3;
    default: n = n+4;
}

Does not compile!
• int x = 5, y = 5, z = 5;
  if (x > 3)
    if (y > 4)
      if (z > 5)
        z += 1;
    else
      z += 2;
    else
      z += 3;
  else
    z += 4;
  z == 11!
Public class Test {
    public static void main(String[] args) {
        private static final int value = 5;
        float total;
        total = value + value / 2;
        System.out.println(total);
    }
}

• Will not compile. Private variables cannot be declared inside a method body.
**this** keyword

- **this** is a pointer to an object’s self
- Always used implicitly, but sometimes useful to be used explicitly
- In case of ambiguity resulting from using the same variable name twice, use **this** to refer to the class’s variable and not the local variable

```java
class A {
    private int a;
    
    public int add(int a) {
        return this.a + a;
    }
}
```

A more clear version avoids ambiguity

```java
class A {
    private int a;
    
    public int add(int b) {
        return a + b;
    }
}
```
this keyword

- Use this keyword to call another constructor

```java
public class A {
    private int x;
    public A() { x=0; }
    public A(int x) { this.x = x; }
    ...
}
```

```java
public class A {
    private int x;
    public A() { this(0); }
    public A(int x) { this.x = x; }
    ...
}
```
Operation Overloading

• It is sometimes useful to have two methods of the same name but accept different parameters (very common in Math library)
• May not differ only by return type or variable name
• Constructors can be overloaded like other methods

Valid:
public String f(int x) {...}
public String f(double x) {...}
public String f(float f) {...}

Invalid:
public int f(int x) {...}
public double f(int x) {...}
Constructors

- Recall: Constructors are used to initialize member variables.
- If you don't write a constructor, Java will provide a default constructor for your class.
- Possible to write multiple constructors.
  - Public A()
  - Public A(int x) // use x as some initial value
  - Public A(A a) // copy constructor
- Providing **any** constructor results in Java not providing the default constructor.
Copy Constructors and Memory Management

• Copy constructors are useful in making a deep copy of an object
  – Shallow copy: simple reference assignment with “=” (what problems may crop up?)
  – Deep copy: copy over all values and objects (deep copy usually) explicitly into new memory
Memory Management

public class A {

    private int x;

    public A() { x=0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public void set(int x) { this.x = x; }
    public int get() { return x; }

}

A a1 = new A();
A a2 = new A(2);
a1 = a2;
a2.set(3);
A a3 = new A(a1);
a1.set(4);

a1.get()?
a2.get()?
a3.get()?
public class A {

    private int x;

    public A() { x=0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public void set(int x) { this.x = x; }
    public int get() { return x; }

    A a1 = new A();
    A a2 = new A(2);
    a1 = a2;
    a2.set(3);
    A a3 = new A(a1);
    a1.set(4);
    a1.get()?4
    a2.get()?4
    a3.get()?3
}
static keyword

- Useful for methods which do not need to access class variables or methods
  - Called using the class name
  - Think: Math library
- Used for class variables which are shared over instances of the class

```java
class A {
    private static int a;

    public void setA(int x) {
        a = x;
    }

    public int getA() {
        return a;
    }
}
```

```java
A a1 = new A();
A a2 = new A();
a1.setA(5);
System.out.println("a: "+a2.getA());
```
public class A {

    private static int x;

    public A() { x=0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public void set(int x) { this.x = x; }
    public int get() { return x; }
}

A a1 = new A();
A a2 = new A(2);
a1 = a2;
a1.set(3);
A a3 = new A(a1);
a1.set(4);
a3.set(1);
a1.get()?
a2.get()?
a3.get()?
static Memory Management

```java
public class A {
    private static int x;

    public A() { x=0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public void set(int x) { this.x = x; }
    public int get() { return x; }
}
```

```
A a1 = new A();
A a2 = new A(2);
a1 = a2;
a1.set(3);
A a3 = new A(a1);
a1.set(4);
a3.set(1);
a1.get()?1
a2.get()?1
a3.get()?1
```
null keyword

• Indicates the non-existence of an object (think address 0x00000000)
• Can be used to represent any object type
• A null object can NOT call any methods
  – This will result in a runtime error called a NullPointerException
  – Get in the habit of checking against the null object!
public class A {
    private int x;
    public A() { x = 0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }
    public void set(int x) { this.x = x; }
    public int get() { return x; }
}

A a1 = null;
a1 = new A();
A a2 = new A(a1);
A a3 = null;
if (a3 == null)
    a3 = new A();
else
    a3.set(3);
a1.set(4);
a1.get()?
a2.get()?
a3.get()?
null Memory Management

public class A {
  private int x;
  public A() { x=0; }
  public A(int x) { set(x); }
  public A(A a) { x = a.get(); }
  public void set(int x) { this.x = x; }
  public int get() { return x; }
}

A a1 = null;
a1 = new A();
A a2 = new A(a1);
A a3 = null;
if (a3 == null)
  a3 = new A();
else
  a3.set(3);
a1.set(4);
a1.get()?4
a2.get()?0
a3.get()?0
Parameters in Java are passed by value (as opposed to pass by reference).

```java
public class A {

    private int x;

    public A() { x = 0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public static void reset(A a) { a = new A(); }
    public void set(int x) { this.x = x; }
    public int get() { return x; }
}

A a1 = new A();
a1.set(4);
A.reset(a1);

a1.get()?
```
Pass-By-Value

Parameters in Java are passed by value (as opposed to pass by reference).

public class A {

    private int x;

    public A() { x=0; }
    public A(int x) { set(x); }
    public A(A a) { x = a.get(); }

    public static void reset(A a) { a = new A(); }
    public void set(int x) { this.x = x; }
    public int get() { return x; }
}

A a1 = new A();
a1.set(4);
A.reset(a1);
a1.get()?4
Javadoc

• Can provide a consistent form for comments in all your Java files
• Can be used to generate API-like html files for your classes
• Starts with /***, ends with */
• Uses @ for special information
  – Author
  – Parameter
  – Return
• As an example, look at the Java 5 API on the class webpage.
Quiz

• What is the difference between an instance variable and a class variable?