

06-0: More on Inheritance

```

class A
{
    public int x;
    protected int y;
    private int z;
}

class B extends A
{
    public int w;
}

A c1 = new A();
B c2 = new B();
A c3 = new B(); /* Show memory contents */

c1.x = 3; ?      c2.x = 3; ?      c3.x = 1; ?
c1.y = 4; ?      c2.y = 4; ?      c3.y = 2; ?
c1.z = 5; ?      c2.z = 5; ?      c3.z = 3; ?
c1.w = 6; ?      c2.w = 6; ?      c3.w = 4; ?

```

06-1: More on Inheritance

```

class A
{
    public int x;
    protected int y;
    private int z;
}

class B extends A
{
    public int w;
}

A c1 = new A();
B c2 = new B();
A c3 = new B(); /* Show memory contents */

c1.x = 3; OK      c2.x = 3; OK      c3.x = 1; OK
c1.y = 4; NOT OK  c2.y = 4; NOT OK  c3.y = 2; NOT OK
c1.z = 5; NOT OK  c2.z = 5; NOT OK  c3.z = 3; NOT OK
c1.w = 6; NOT OK  c2.w = 6; OK      c3.w = 4; NOT OK

```

06-2: Yet More on Inheritance

```

class A
{
    public int x;
    protected int y;
    private int z;

    void set3(int a, int b, int c)
    {
        x = a;
        y = b;
        z = c;
    }
}

class B extends A
{
    private int w;

    void set4(int a, int b, int c, int d)
    {
        x = a; ?
        y = b; ?
        z = c; ?
        w = d; ?
    }
}

A c1 = new A();
A c2 = new B();

c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10);

```

06-3: Yet More on Inheritance

```

class A
{
    public int x;
    protected int y;
    private int z;

    void set3(int a, int b, int c)
    {
        x = a;
        y = b;
        z = c;
    }
}

class B extends A
{
    private int w;

    void set4(int a, int b, int c, int d)
    {
        x = a; OK
        y = b; OK
        z = c; BAD (private!)
        w = d; OK
    }
}

A c1 = new A();
A c2 = new B();

c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10);

How could we do this?

```

06-4: Yet More on Inheritance

```

class A
{
    public int x;
    protected int y;
    private int z;

    void set3(int a, int b, int c)
    {
        x = a;
        y = b;
        z = c;
    }
}

class B extends A
{
    private int w;

    void set4(int a, int b, int c, int d)
    {
        set3(a, b, c);
        w = d;
    }
}

A c1 = new A();
A c2 = new B();

c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10);

```

06-5: Yet More on Inheritance

```

class A
{
    public int x;
}

class B extends A
{
    public int d;
}

In main:
-----
A a;
B b;

a.x = 3; OK?
b.x = 4; OK?
b.d = 5; OK?

```

06-6: Yet More on Inheritance

```

class A
{
    public int x;
}

class B extends A
{
    public int d;
}

In main:
-----
A a;
B b;

a.x = 3; NOT OK -- cannot use class variable until "new" is called
All class variables (and arrays!) are stored on the heap
b.x = 4; NOT OK
b.d = 5; NOT OK

```

06-7: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}

class B extends A
{
    public int d;
}

class C
{
    public int e;
}

In Main
-----
A a = new A();
B b = new B();

a.x = 3; OK?
a.c.e = 4; OK?
b.x = 5; OK?
b.d = 6; OK?
a.intArray[2] = 4; OK?

```

06-8: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}

class B extends A
{
    public int d;
}

In Main
-----
A a = new A();
B b = new B();

a.x = 3; OK
a.c.e = 4; NOT OK
b.x = 5; OK
b.d = 6; OK

```

```

{
    public int d;
}
class C
{
    public int e;
}
a.intArray[2] = 4; NOT OK

```

06-9: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}
class B extends A
{
    public int d;
}
class C
{
    public int e;
}
In Main
-----
A a = new A();
A a2 = new A();
a.cArray = new C[5]; OK?
a.cArray[2].e = 5; OK?
a2.e = 4; OK?
a2.intArray = new int[5]; OK?
a2.intArray[2] = 4; OK?

```

06-10: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}
class B extends A
{
    public int d;
}
class C
{
    public int e;
}
In Main
-----
A a = new A();
A a2 = new A();
a.cArray = new C[5]; OK
a.cArray[2].e = 5; NOT OK
a2.e = 4; NOT OK
a2.intArray = new int[5]; OK
a2.intArray[2] = 4; OK

```

06-11: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}
class B extends A
{
    public int d;
}
class C
{
    public int e;
}
In Main
-----
A a[] = new A[2];
int x;
int y[] = new int[3];
a[2].x = 4 OK?
a[2].cArray = new C[3]; OK?
a[2].cArray[2].e = 4 OK?
a.x = 3 OK?
x = 3; OK?
y[3] = 4; OK?

```

06-12: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}
class B extends A
{
    public int d;
}
class C
{
    public int e;
}
In Main
-----
A a[] = new A[2];
int x;
int y[] = new int[3];
a[2].x = 4 NOT OK
a[2].cArray = new C[3]; NOT OK
a[2].cArray[2].e = 4 NOT OK
a.x = 3 NOT OK
x = 3; OK
y[3] = 4; OK

```

06-13: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}

class B extends A
{
    public int d;
}

class C
{
    public int e;
}

In Main
-----
A a[] = new A[3];

a[0] = new B();   OK?
a[1] = new A();   OK?
a[2] = new B();   OK?
a[3] = new A();   OK?
a[0].x = 2;       OK?
a[a[0].x].x = 4   OK?
a[2].e = 5;       OK?
a[2].c = new C(); OK?
a[2].c.e = 6;     OK?

```

06-14: Yet More on Inheritance

```

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
}

class B extends A
{
    public int d;
}

class C
{
    public int e;
}

In Main
-----
A a[] = new A[3];

a[0] = new B();   OK
a[1] = new A();   OK
a[2] = new B();   OK
a[3] = new A();   OK
a[0].x = 2;       OK
a[a[0].x].x = 4   OK
a[2].e = 5;       NOT OK
a[2].c = new C(); OK
a[2].c.e = 6;     OK

```

06-15: Interior Classes

```

class A {
    public int x, y;
    public B a,b;

    public A() {
        a = new B();
        b = new B();
    }

    class B {
        public int w;
        void foo() {
            x = y + w;
        }
    }

    void bar() {
        a.w = 3;
        b.w = 4;
        a.foo();
        b.foo();
    }
}

in main:
A outer = new A();
outer.x = 3;
outer.bar();
System.out.println(outer.y);
outer.x = 5;
System.out.println(outer.y);

```

06-16: Panel Layout

- When elements are added to panels, done from left to right
- When there is not enough room to right, wrap around
- Hard to get exactly the layout you want
- Layout can change drastically when window resizes
- What can we do?

06-17: Panel Layout

- Can add a “layout manager” to panel
- Determines how elements (buttons, labels, subpanels, etc) are added to panel

- There are several different kinds of layout managers
 - BorderLayout
 - GridLayout

06-18: Panel Layout

```
JFrame f = new JFrame("Window");
JPanel p = new JPanel();
f.add(p);

p.setLayout(new BorderLayout());
JLabel l1 = new JLabel("North");
JLabel l2 = new JLabel("South");
JLabel l3 = new JLabel("East");
JLabel l4 = new JLabel("West");
JLabel l5 = new JLabel("Center");
p.add(l1, BorderLayout.NORTH);
p.add(l2, BorderLayout.SOUTH);
p.add(l3, BorderLayout.EAST);
p.add(l4, BorderLayout.WEST);
p.add(l5, BorderLayout.CENTER);
```

06-19: Panel Layout

- We get a slightly different behavior if we nest panels using the BorderLayout
 - (See other file)

06-20: Panel Layout

- GridLayout
 - Allows us to add elements to panel in a grid
 - (will be very useful for Project 2!)

```
JFrame f = new JFrame("Layout test");
JPanel p = new JPanel();
f.add(p);
p.setLayout(new GridLayout(2,3));

p.add(new JButton("1"));
p.add(new JButton("2"));
p.add(new JButton("3"));
p.add(new JButton("4"));
p.add(new JButton("5"));
p.add(new JButton("6"));

f.pack();
f.setVisible(true);
```

06-21: Panel Layout

- Get your hands dirty!
 - Create a window that looks like this (buttons can be non-functional, label should be centered)