Name
Advanced Programming in Java
Lab Exercise 12/3/2020

Lesson 36 - Inheritance

```
public class Red extends Green
{
        public int blue(double x)
        { . . . }
        public String s;
        private int i;
}

public class Green
{
        public double peabody(double y)
        {
                return mm;
        }
        private boolean crunch( )
        { . . . }
        private double mm;
        public long xx;
}
```

1. Which of the above two classes is the base class?

Green

2. Which of the above two classes is the subclass?

Red

3. Which of the above two classes is the superclass?

Green

4. Which of the above two classes is the derived class?

Red

5. Is this legal? If not, why not? (Assume this code is in some class other than the two above)

```
Red myObj = new Red( );
boolean bb = myObj.crunch( );
```

No, *crunch* is *private*; otherwise, if *crunch* was *public* it would be ok.

6. Is this legal? If not, why not? (Assume this code is in some class other than the two above)

```
Red myObj = new Red( );
int bb = myObj.blue( 105.2);
```

Yes

7. Write code for the *blue* method that will printout the *mm* state variable.

System.out.println(peabody(37.2));

8. Write code for the *blue* method that will printout the *xx* state variable.

System.out.println(xx);

Use the following two classes for problems 9 - 12:

```
public class Red extends Green
{
        public int blue(double x)
        { . . . }
        public double peabody(double vv)
        { . . . }
        public String s;
        private int i;
}

public class Green
{
        public Green(long j)
        {
                xx = j;
        }
        public double peabody(double y)
        {
                return mm;
        }
        private Boolean crunch( )
        { . . . }
        private double mm;
        public long xx;
}
```

9. Consider the following constructor in the *Red* class:

```
public Red( )
{
        //What code would you put here to invoke the constructor in the
        //superclass and send it a parameter value of 32000?
}
```

super(32000);

10. Is there any method in *Red* that is overriding a method in *Green*? If so, what is it?

Yes, *Peabody*.

11. Look at the *peabody* method inside *Red*. Write the code inside that method that will allow you to access the same method inside its superclass, *Green*. Let the parameter have a value of 11.

double xxx = super.peabody(11);

12. Consider the following constructor in the *Red* class:

```
public Red( )
{
        String s = "Hello";
        super(49);
}
```

Is this code legal? If not, why not?

No, *super(49)* should be the first line of code in this constructor.

13. Assume that the following fragments of code are all in a subclass. Match each to an item from the "sentence bank" to the right.

__b__ this.(x,y)	a. refers to a constructor in the superclass
__d___ this.z	b. refers to a constructor in the subclass
__a___ super(j)	c. refers to an overridden method in the super class
__c___ super.calc( )	d. refers to a data member in the subclass

Exercise (B) on Lesson 36

The following code applies to problems 1 - 3:

```
public abstract class Hammer
{
        public abstract void duty( );
        public abstract int rule(int d);
}

public class Lurch extends Hammer
{
        public void duty( )
        {
                int x = Y;
        }
        public int rule( int d)
        {
                Y = d + 1;
        }
        private int Y = 30;
        private int x;
}
```

1. What is the purpose of making the two methods above abstract?

    To force implementation in whatever inherits *Hammer* (*Lurch* in this case)

2. Write out the full signature of the *rule* method.

    public int rule(int d)

3. Which class actually implements the *duty* method?

    Lurch

4. A class for which you cannot create objects is called a (an)_abstract___ class.

5. Given:

```
public abstract class Felix
{
. . .
}
```

Is the following attempt at instantiating an object from the *Felix* class legal? If not, why?

```
Felix myFelix = new Felix( );
```

No, *Felix* is abstract

6. Is the following legal? If not, why?

```
public abstract class Lupe
{
        public abstract void fierce( )
        { . . . }
        public final double PI = 3.14;
}
```

No, semicolon should follow fierce(), and can't have a body for an abstract method.

7. What is the main reason for using abstract classes?

To prevent objects from being made from the class

8. Modify the following class so it is impossible to make subclasses from it.

```
public class MyClass
{
. . .
}
```

```
public final class MyClass
{
. . .
}
```

9. Why would the following code be pointless?

```
public final abstract class MyClass
{
        . . .
        //there are no static methods
}
```

If you can't inherit (final) or create objects (abstract), it's useless…unless it has static methods.

10. Given:

```
public class ChevyChase
{
        public void Chicago(int x)
        {
                . . .
        }
}
```

Modify the above code so as to make it impossible for a subclass that extends *ChevyChase* to override the *Chicago* method.

```
public class ChevyChase
{
        public final void Chicago(int x)
        {
                . . .
        }
}
```

11. Is it possible to override instance fields (also called state variables)?

No

12. What is shadowing (as the term applies to superclasses and subclasses)?

Having state variables of the same name in both superclass and subclass

The following code applies to problems 13 – 14, 18 - 20:

```
public class Parent
{
        public void rubyDoo( )
        { . . . }
        • • •
        public int x = 0;
}

public class Child extends Parent
{
        public void busterStein( )
        { . . . }
        • • •
        public int x = 39;
}
```

13. Consider the following code in a *Tester* class:

```
Child myChild = new Child( );
System.out.println(myChild.x); //What gets printed?
```

39

14. Consider the following code in a *Tester* class:

```
Child myChild = new Child( );
```

Is there any way using the *myChild* object to retrieve the *x* state field within the *Parent* class? Write the code that will do this. You may write a new method for either class if you need to.

In *Tester*, add… int i = myChild.get_x( );

In the Parent class add the following method:

```
public int get_x( )
{
        return x;
}
```

15. What is the name of the Cosmic Superclass?

Object

16. What is the name of the class that every class (that does not extend another class) automatically extends?

Object

17. What are the three main methods of the *Object* class?

toString, equals, clone

18. Is the following legal? If not, why not?

```
Child theObj = new Child( );
Parent newObj = theObj;
newObj.busterStein( );
```

No, *busterStein* belongs only to the *Child* class and is not accessible to this *Parent* object

19. Is the following legal? If not, why not?

```
Child theObj = new Child( );
Parent newObj = theObj;
newObj.rubyDoo( );
```

Yes

20. Is the following legal? If not, why not?

```
Parent meatloaf = new Child( );
```

Yes

For problems 21-25, consider the following. In each problem either state what is printed or indicate that it won't compile:

```
public class A
{
        public A (int x)
        {
                this.x = x;
        }
        public int f( )
        {
                return x;
        }
        public int g( )
        {
                return x;
        }
        public int x;
}

public class B extends A
{
        public B (int x, int y)
```

```
        {
                super(x);
                this.x = y;
        }

        public int f( )
        {
                return x + g( );
        }
        public int zorro( )
        {
                return x + g( );
        }
        public int x;
}
```

21. A a = new B(5, 10);
System.out.println(a.g( ));

5

22. A a = new B(5, 10);
System.out.println(a.f( ));

15… the *f* method in *B* is used because it overrides *f* in the *A* class

23. A a = new B(5, 10);
System.out.println(a.x);

5…note we don't use the *x* in the *B* class since state variables can't be overridden.

24. B a = new B(5, 10);
System.out.println(a.x);

10

25. A a = new B(5, 10);
System.out.println(a.zorro( ));

Won't compile, *zorro* is not part of the *A* class

26. Consider the classes *Food*, *Cheese*, and *Velveta* where *Cheese* is a subclass of *Food* and *Velveta* is a subclass of *Cheese*. State which of the following lines of code are legal.

Cheese c = new Food( );	_____
Velveta v = new Food( );	_____
Cheese c = new Velveta( );	___legal_____
Food f = new Velveta( );	___legal_____
Food f = new Cheese( );	___legal_____