AP CS Unit 6: Inheritance Exercises

1. Suppose your program contains two classes: a Student class and a Female class. Which of the following is true?

a) Making the Student class a subclass of the Female class is a good design decision.
b) Making the Female class a subclass of the Student class is a good design decision.
c) Either class could be a subclass of the other class; it would still be a good design decision.
d) The only good design decision would be to make neither a subclass of the other.

2. Suppose your program contains two classes: a Rectangle class and a Square class. Which of the following is true?

a) Making the Rectangle class a subclass of the Square class is a good design decision.
b) Making the Square class a subclass of the Rectangle class is a good design decision.
c) Either class could be a subclass of the other class; it would still be a good design decision.
d) The only good design decision would be to make neither a subclass of the other.

3. Suppose your program contains two classes: a Computer class and a Monitor class. Which of the following is true?

a) Making the Computer class a subclass of the Monitor class is a good design decision.
b) Making the Monitor class a subclass of the Computer class is a good design decision.
c) Either class could be a subclass of the other class; it would still be a good design decision.
d) The only good design decision would be to make neither a subclass of the other.

Use the java api 7 to answer questions 4 to 7.

4. What is the superclass of the Math class? _________________________

5. The JButton class is the subclass of the_______________________class which is the subclass of the_______________________class which is the subclass of the ________________________ class which is the subclass of the________________________which is the subclass of the Object class. (Be sure to look at how many methods JButton inherits.)

6. What is the data type of the variable *out*? ___________________
Go to that class and look up the print method. The print method is overloaded. How many different print methods are there? ________
	Cow c = new Cow();
	System.out.print(c);

7. Select the TRUE statement(s). a) The print method displays the contents of *c* which is a reference to a Cow object. b) If the argument to the print method is an object, the object's toString method will be called and the string it returns will be displayed. c) If the Cow class did not contain a toString method then the code will either not compile or it will crash at run-time.	

```
public class Animal{
        public void m1(){
                System.out.print("A");
        }
        public void m2(){
                System.out.print("B");
        }
}
```

```
public class Dog extends Animal{
        public void m1(){
                System.out.print("C");
        }
        public void m3(){
                System.out.print("D");
        }
}
```

Problems 8 to 13 refer to the above classes. If it does not compile, write COMPILER ERROR.

8. What is displayed?	Dog d = new Dog();
	d.m1();

9. What is displayed?	Dog d = new Dog();
	d.m2();

10. What is displayed?	Dog d = new Dog();
	d.m3();

11. What is displayed?	Animal a = new Animal();
	a.m3();

12. Does this compile and will it run?	Dog x = new Dog();
	String s = x.toString();

13. The m1 method in the Dog class overrides the m1 method in the Animal class.	TRUE FALSE

```
public class Vehicle{
    public void v(){
        SOP("V");
    }
    public void e(){
        SOP ("E");
    }
}
```

```
public class Plane extends Vehicle{
    public void p(){
        SOP("P");
    }
    public void v(){
        super.v();
        SOP("X");
    }
}
```

```
public class Jet extends Plane{
    public void j(){
        SOP("J");
    }
    public void e(){
        SOP("A");
        super.e();
    }
}
```

Note. Of course SOP is not valid code. I'm using an abbreviation to save space.

Problems 14 to 20 refer to the above classes. If it does not compile, write COMPILER ERROR.

14. What is displayed?	Vehicle k = new Vehicle();
	k.p();

15. What is displayed?	Plane k = new Plane();
	k.v();

16. What is displayed?	Plane k = new Plane();
	k.e();

17. What is displayed?	Jet k = new Jet();
	k.e();

18. What is displayed?	Jet k = new Jet();
	k.v();

19. Does this compile and will it run?	Jet z = new Jet();
	boolean b = z.equals("?");

20. Ignoring methods inherited from the Object class, what methods could be called in the second line?	Jet z = new Jet(); z.____________

```
public class Toy{
        public Toy(){
                System.out.print("T");
        }
        public Toy( String s ){
                System.out.print( s );
        }
}
```

```
public class YoYo extends Toy{
        public YoYo(){
                super("A");
                System.out.print("Y");
        }
        public YoYo(String s){
                System.out.print(s);
        }
}
```

Problems 21 to 24 refer to the above classes. These all compile and run.

21. What is displayed?	`YoYo y = new YoYo();`
22. What is displayed?	`YoYo y = new YoYo( "W" );`
23. The constructors in the Toy class are overloaded.	TRUE FALSE
24. The second constructor in the YoYo class overrides the second constructor in the Toy class.	TRUE FALSE

```
public class Tool{
        private int x;

        public Tool(int n){
                x = n;
        }
        public int get(){
                return x;
        }
}
```

Problems 25 to 27 refer to the above class.

25. There is a compiler error in this constructor. Fix it. And don't touch the Tool class; that's perfect. Fix the Hammer class's constructor.

```
public class Hammer extends Tool{
        private int y;

        public Hammer(){
                y = 7;
        }
}
```

26. Select the TRUE statement.
a) The Axe class compiles.
b) Line 1 causes a compiler error; line 2 is fine.
c) Line 2 causes a compiler error; line 1 is fine.
d) Lines 1 and 2 both cause compiler errors.

```
  public class Axe extends Tool{
          public Axe(){
1                 super( 4 );
2                 x = 8;
          }
  }
```

27. The Saw class compiles.

What is the value of *n*?

```
// client code

Saw jig = new Saw( 3, 8 );
int n = jig.get();
```

```
public class Saw extends Tool{
     private int x;

     public Saw( int a, int b ){
          super( a );
          x = b;
     }
}
```

28. Only one of the two ShoeBox constructors compile. Which one does NOT compile and why?

```
public class Box {
      private int x;

      public void set(int n){
            x = n;
}
```

```
public class ShoeBox extends Box{
      private int z;

      public ShoeBox (int h) {
            z = h;
            x = h;
      }

      public ShoeBox () {
            z = 8;
            set( 8 );
      }
}
```

```
public class Plant {
      public void m1(){
            SOP("P1");
      }
      public void m2(){
            SOP("P2");
            m1();
      }
      public void m8(){
            SOP("P8");
      }
}
```

```
public class Tree extends Plant{
      public void m1(){
            SOP("T1");
      }
      public void m3(){
            m1();
            SOP("T3");
      }
      public void m8(){
            super.m8();
            SOP("T8");
      }
}
```

```
public class Elm extends Tree{
      public void m1(){
            SOP("E1");
      }
      public void m4(){
            SOP("E4");
      }
      public void m8(){
            SOP("E8");
            super.m8();
      }
}
```

SOP is an abbreviation for System.out.print.
Problems 29 to 35 refer to the above classes

29. What is displayed?	`Plant p = new Plant();` `p.m2();`
30. What is displayed?	`Tree t = new Tree();` `t.m3();`
31. Will this compile and run? If yes, what is displayed? (There is a lot to be learned by this problem.)	`Tree t = new Tree();` `t.m2();`
32. Will this compile and run? If yes, what is displayed?	`Elm e = new Elm();` `e.m3();`
33. Will this compile and run? If yes, what is displayed?	`Elm e = new Elm();` `e.m2();`
34. What is displayed?	`Tree t = new Tree();` `t.m8();`
35. What is displayed?	`Elm e = new Elm();` `e.m8();`

36. This code generates the following compiler error:
toString() in Moe cannot override toString() in java.lang.Object; attempting to use incompatible return type

Explain what this means.

```
public class Moe {
   public void toString() {
        // code
   }
}
```

37. This class compiles but its equals method does not override the equals method in the Object class. Explain why it doesn't.

```
public class AA {
   public boolean equals( int n ){
        return true;
   }
}
```

38. This does compile and run. It displays …
a) true
b) false

```
String s = "??";
System.out.println( s instanceof Object );
```

39. Why does the first line compile?

40. What is displayed?

```
Object x = "??";
System.out.println( x instanceof String );
System.out.println( x instanceof Integer );
System.out.println( x instanceof Math );
```

41. Name two reasons why the second line does not compile.

```
int z = 13;
System.out.println( z instanceof int );
```

42. Why is it ok to pass a String or Integer object to a method that has a parameter of type Object?

43. Circle the one line that causes the following run-time error: ClassCastException

44. Could the body of method2 be replaced with this one line:
String e = c.toString();
and still produce the same results?

```
public class Runner {
    public static void main(String[] args) {
        method1( "hi" );
        Integer h = new Integer( 15 );
        method2( h );
    }

    public static void method1(Object a){
        Integer b = (Integer) a;
    }

    public static void method2(Object c){
        Integer d = (Integer) c;
        String e = d.toString();
    }
}
```

```
public class Clock{
    private int x;

    public Clock(int n){
        x = n;
    }

    public boolean equals(Object x){
        if (x instanceof Clock) {
            Clock c = (Clock)x;
            return this.x == c.x;
        }else
            return false;
    }
}
```

```
public class Cuckoo extends Clock{
    private int y;

    public Cuckoo(int a, int b){
        super(a);
        y = b;
    }

    public boolean equals(Object x){
        if (!(x instanceof Cuckoo) )
            return false;

        Cuckoo c = (Cuckoo)x;
        if ( super.equals(c) && y == c.y )
            return true;
        else
            return false;
    }
}
```

45. What is displayed?

```
Clock c1 = new Clock( 12 );
Cuckoo c2 = new Cuckoo( 12, 9 );
Cuckoo c3 = new Cuckoo( 12, 9 );
System.out.println( c1.equals( c2 ) );
System.out.println( c2.equals( c1 ) );
System.out.println( c3.equals( c2 ) );
```

Note. There general guidelines for writing the equals method. One rule is that if A equals B then B should equal A. I broke this rule for the purposes of keeping the code simple.

```
public class Person{
        private String name;

        public Person( String s ){
                name = s;
        }

        public String toString(){
                return name;
        }
}
```

```
public class Student extends Person {
        private int grade;

        public Student( int g, String s ){
                super( s );
                grade = g;
        }

        public String toString(){
                String x = super.toString();
                return x + " grade " + grade;
        }
}
```

46. What is displayed?

```
Person [] p = new Person[ 4 ];
p[0] = new Person( "Miller");
p[1] = new Student( 12, "Friend" );
p[2] = new Student( 10, "Chheu" );
p[3] = new Person( "Liu" );

for ( int k = 0; k < p.length; k++ )
        System.out.println( p[k] );
```

```
public class TryIt {
    public static void main(String [] args) {
        ABC bob = new ABC();
        XYZ jill = new XYZ();

        System.out.println(bob.methodD(10));

        System.out.println(bob.methodE(20.0) );

        System.out.println(jill.methodD(30) );

        System.out.println(jill.methodE(40) );

        System.out.println(jill.methodE(50.0) );

    }
}
```

47) What is displayed when the above main method is executed?

____

____

____

____

____

```
public class ABC {
    public int methodD(int x) {
        return 2*x;
    }

    public double methodE(double x) {
        return 3*x;
    }

}



// another file
public class XYZ extends ABC {
        public int methodD(int x) {
                int y = super.methodD(x);
                return 5*y;
        }

        public double methodE(int x) {
                double y = methodE(0.5*x);
                return 7*y;
        }
}
```

48) What does the client code display? If there is a compiler or run-time error, write ERROR. Pretend that earlier errors do not effect later statements.

```
// client code

BB x = new BB();

x.m1(); _______________

x.m2(); _______________

x.m3(); _______________


AA y = new BB();

y.m1();_______________

y.m2();_______________

y.m3();_______________
```

```
public class AA {
public void m1() {
   System.out.print("A1");
   m2();
}

public void m2() {
   System.out.print("A2");
}
}
```

```
public class BB extends AA {

public void m2() {
   System.out.print("B2");
}

public void m3() {
   System.out.print("B3");
}
}
```

49. This does compile and run. Select the TRUE statement.
a) It calls the equals method defined in the Object class.
b) It calls the equals method defined in the String class.

```
Object x = "cold";
Object y = "hot";
System.out.println(x.equals(y));
```

50. This compiles and runs. What is displayed?

```
// client code
Object [] a = new Object[5];
a[0] = "cat";
a[1] = new Book();
a[2] = new Book();
a[3] = "dog";

for ( int i = 0; i < 5; i++ )
     System.out.println( a[i] );
```

```
public class Book {
    public String toString(){
          return "book";
    }
}
```

```
public class Planet{
        public int get(){
                return 4;
        }
}
```

```
public class Earth extends Planet {
        public int get(){
                return 8;
        }
}
```

51. What is x?

```
Planet p = new Earth();
int x = p.get();
```

```
public class Kennel {
    public static void main(String[] args) {

        Dog emma = new Dog( 12 );

        Dog king = new Dog( 12 );

        System.out.println( emma == king );

        System.out.println( emma.equals(king) );
    }
}
```

52. The above runs and prints:

____

____

```
public class Dog {
   private int age = 0;

   public Dog( int a ) {
       age = a;
   }

   public boolean equals( Object obj ){
       if ( !(obj instanceof Dog) )
            return false;

       if ( age == ((Dog) obj).age )
            return true;
       else
            return false;
   }
}
```

53. Assume this compiles and runs. Do the two methods behave exactly the same?

a) Yes

b) No

```
public void m1( Monster m ) {
     if (m instanceof Ghost ) ){
          Ghost g = (Ghost) m;
          g.dance();
     } else
          m.dance();
}
```

```
public void m1( Monster m ) {
     m.dance();
}
```

```
public class MyMain {
    public static void main( String [] args ) {
        XX x = new XX();
        x.methodA();
        x.methodB();

        System.out.println( "***********" );
        YY y = new YY();
        y.methodA();
        y.methodB();
    }
}
```

54. When the above main method is run, 11 lines are displayed. Fill in the 10 blanks. The second half is very tricky but it is one that you need to understand.

____

____

____

____

```
***********
```

____

____

____

____

____

____

```
public class XX {
        public XX() {
                System.out.println( "X" );
        }

        public void methodA() {
                System.out.println( "A" );
        }

        public void methodB() {
                methodA();
                System.out.println( "B" );
        }
}

// another file
public class YY extends XX{
        public YY() {
                System.out.println( "Y" );
        }

        public void methodA() {
                System.out.println( "AX" );
        }

        public void methodB() {
                super.methodB();
                System.out.println( "BX" );
        }
}
```

55. Will this method compile and run? If yes, describe what it does. If no, what is the problem?

```
public int county( Object[] list ){
        int n = 0;
        for ( int i = 0; i < list.length; i++ ){
                if ( list[i] instanceof String )
                        n++;
        }
        return n;
}
```

56. Select the TRUE statement(s).
a) Machine may be an abstract class.
b) The Machine class must define or inherit a process method.
c) The Robot class may or may not define a process method.
d) Robot cannot be an abstract class.

```
// assume this compiles/runs
Machine x = new Robot();
x.process();
```

57. Will this method compile? If no, explain.

If it compiles, will it run error free? Explain.

```
public int county( Object[] list ){
        int n = 0;
        for ( int i = 0; i < list.length; i++ ){
                String str = (String) list[i];
                n += str.length();
        }
        return n;
}
```

58. Select the FALSE statement.
   a) Every method in an abstract class is an abstract method.
   b) Every class inherits methods from the Object class.
   c) A private method in a superclass cannot be invoked by a subclass.
   d) A concrete class that extends an abstract class must override any abstract methods.

```
public class Runner{
    public static void main(String [] args) {
        Tuna tina = new Tuna();
        Fish f = new Tuna();
        System.out.println( tina.swim() );
        System.out.println( tina.breathe() );
        System.out.println( f.breathe() );
        System.out.println( f.cook() );
    }
}
```

59. What is displayed when the above main method is executed? One of the statements generates a compiler error. Write ERROR for that line and pretend it doesn't affect the others.

____

____

____

____

```
public class Fish {
    public String swim() {
        return "a";
    }

    public String breathe() {
        return "b";
    }
}

// another file
public class Tuna extends Fish {
     public String breathe() {
         String s = "";
         for ( int k=1; k<=3; k++ )
              s += super.breathe();
         return s;
     }

     public String cook() {
         return "yum";
     }
}
```

60. For this to compile, Lamp should be a subclass of Light and the Light class must define an *on* method.
TRUE   FALSE

61. At runtime the JVM will take the reference in *x* and try to run the *on* method in the Lamp class. If Lamp does not define an *on* method then it will use the *on* method from the Light class.
TRUE   FALSE

```
Light x = new Lamp();
x.on();
```

62. If this compiles then Monkey cannot be an abstract class. TRUE FALSE	Monkey m = new Monkey();
63. If this compiles then Animal cannot be an abstract class. TRUE FALSE	Animal a = new Bear();

64. Does this class compile?

```
public abstract class Polygon {
        private boolean convex;

        public Polygon( boolean b ){
                convex = b;
        }

        public abstract double getArea();

        public boolean get(){
                return convex;
        }
}
```

65. Write the Square class which is a non-abstract subclass of the Polygon class. The Square should have one instance variable, int side, which represents the length of one side. The Square constructor should have one parameter that is used to initialize the instance variable.

And yes, the above Polygon class does compile and so should your Square class.

66. How many Thing objects does this statement create? 67. If this compiles then Thing cannot be an abstract class. TRUE FALSE	Thing [] m = new Thing[20];