AP Computer Science
Review for Chapter 9 Test

1 Multiple-Choice Questions

For the questions#1 ~ 4 below, use the following partial class definitions:

```
public class A1 {
      public int x;
      private int y;
      protected int z;
...
}

public class A2 extends A1 {
      protected int a;
      private int b;
...
}

public class A3 extends A2 {
      private int q;
...
}
```

1) Which of the following is true with respect to `A1`, `A2` and `A3`?
A) `A1` is a subclass of `A2` and `A2` is a subclass of `A3`
B) `A3` is a subclass of `A2` and A2 is a subclass of `A1`
C) `A1` and `A2` are both subclasses of `A3`
D) `A2` and `A3` are both subclasses of `A1`
E) `A1`, `A2` and `A3` are all subclasses of the class `A`

Answer: B

2) Which of the following lists of instance data are accessible in class `A2`?
A) `x, y, z, a, b`
B) `x, y, z, a`
C) `x, z, a, b`
D) `z, a, b`
E) `a, b`

Answer: C

3) Which of the following lists of instance data are accessible in `A3`?
A) `x, y, z, a, b, q`
B) `a, b, q`
C) `a, q`
D) `x, z, a, q`
E) `x, a, q`

Answer: D

4) Which of the following is true regarding the use of instance data y of class A1?

A) It is accessible in A1, A2 and A3
B) It is accessible in A1 and A2
C) It is accessible only in A1
D) It is accessible only in A3
E) It is not accessible to any of the three classes

Answer: C

5) The instruction `super( );` does which of the following?
A) calls the method super as defined in the current class
B) calls the method super as defined in the current class'parent class
C) calls the method super as defined in java.lang
D) calls the constructor as defined in the current class
E) calls the constructor as defined in the current class'parent class

Answer: E

6) Aside from permitting inheritance, the visibility modifier *protected* is also used to
A) permit access to the protected item by any class defined in the same package
B) permit access to the protected item by any static class
C) permit access to the protected item by any parent class
D) ensure that the class can not throw a NullPointerException
E) define abstract elements of an interface

Answer: A

7) Which of the following is an example of multiple inheritance?
A) A computer can be a mainframe or a PC
B) A PC can be a desktop or a laptop
C) A laptop is both a PC and a portable device
D) A portable device is a lightweight device
E) Macintosh and IBM PC are both types of PCs

Answer: C

8) Java does not support multiple inheritance, but some of the abilities of multiple inheritance are available by
A) importing classes
B) implementing interfaces
C) overriding parent class methods
D) creating aliases
E) using public rather than protected or private modifiers

Answer: B

9) Abstract methods are used when defining
A) interface classes
B) derived classes
C) classes that have no constructor
D) arrays
E) classes that have no methods

Answer: A

For the questions #10 ~ 11 below, consider the following class definition:

```
public class AClass {

      protected int x;
      protected int y;

      public AClass(int a, int b) {
            x = a;
            y = b;
      }

      public int addEm( ) {
            return x + y;
      }

      public void changeEm( ) {
            x++;
            y--;
      }

      public String toString( ) {
            return "" + x + "   " + y;
      }
}
```

10) Consider that you want to extend `AClass` to `BClass`. `BClass` will have a third int instance data, z. Which of the following would best define BClass'constructor?

A)
```
public BClass(int a, int b, int c)
{
      super(a, b, c);
}
```

B)
```
public BClass(int a, int b, int c)
{
      x = a;
      y = b;
      z = c;
}
```

C)
```
public BClass(int a, int b, int c)
{
      z = c;
}
```

D)
```
public BClass(int a, int b, int c)
{
      super(a, b);
      z = c;
}
```

E)
```
public BClass(int a, int b, int c)
{
      super( );
}
```

Answer: D

11) You want `addEm` to now add all three values and return the sum and changeEm to change `x` and `y`, but leave `z` alone. Which should you do?
A) Redefine `addEm` and `changeEm` without referencing `super.addEm( )` or `super.changeEm( )`
B) Redefine `addEm` to return the value of z + `super.addEm( )`, but leave `changeEm` alone
C) Redefine `changeEm` to call `super.changeEm( )` and then set `z = x + y`, but leave `addEm` alone
D) Redefine `addEm` to return the value of `z + super.addEm( )` and redefine `changeEm` to call `super.changeEm( )` and then set `z = x + y`
E) Redefine `changeEm` to call `super.changeEm( )` without doing anything to `z`, and redefine `addEm` to return `super.addEm( )`

Answer: B

12) Two children of the same parent class are known as
A) aliases
B) relatives
C) clones
D) brothers
E) siblings

Answer: E

13) In order to determine the type that a polymorphic variable refers to, the decision is made
A) by the programmer at the time the program is written
B) by the compiler at compile time
C) by the operating system when the program is loaded into memory
D) by the Java run-time environment at run time
E) by the user at run time

Answer: D

For the questions #14 ~ 16 below, assume that Student, Employee and Retired are all extended classes of Person, and all four classes have different implementations of the method getMoney. Consider the following code where ... are the required parameters for the constructors:

```
Person p = new Person(...);
int m1 = p.getMoney( );          // assignment 1
p = new Student(...);
int m2 = p.getMoney( );          // assignment 2
if (m2 < 100000) p = new Employee(...);
  else if (m1 > 50000) p = new Retired(...);
int m3 = p.getMoney( );          // assignment 3
```

14) The reference to `getMoney( )` in assignment 1 is to the class
A) `Person`
B) `Student`
C) `Employee`
D) `Retired`
E) none of the above, this cannot be determined by examining the code

Answer: A

15) The reference to `getMoney( )` in assignment 2 is to the class
A) Person
B) Student
C) Employee
D) Retired
E) none of the above, this cannot be determined by examining the code

Answer: B

16) The reference to `getMoney( )` in assignment 3 is to the class
A) Person
B) Student
C) Employee
D) Retired
E) none of the above, this cannot be determined by examining the code

Answer: E

17) The relationship between a parent class and a child class is referred to as a(n) _____ relationship.
A) has-a
B) is-a
C) was-a
D) instance-of
E) alias

Answer: B

For the questions #18 ~ 19, consider the following class definition:

```
public class Q18_19 {
      private int x;

      public Q18_19(int newValue) {
            x = newValue;
      }
}
```

18) Which of the following is true about the class `Q18_19`?
A) It has no parent class
B) Its parent class is Object
C) Its parent class is Java
D) It can not be extended
E) It has a default child called Object

Answer: B

19) If `q1` and `q2` are objects of `Q18_19`, then `q1.equals(q2)`
A) is a syntax error since equals is not defined in the `Q18_19` class
B) is true if `q1` and `q2` both store the same value of `x`
C) is true if `q1` and `q2` reference the same `Q18_19` object
D) is never true
E) throws a NullPointerException

Answer: C

20) Which of these is correct?
A) a base class is a parent class or super class
B) a base class is a child class or derived class
C) a child class is a super class of its parent
D) a parent class is a subclass of its child
E) none of the above

Answer: A

21) Using the reserved word, `super`, one can
A) access a parent class'constructor(s)
B) access a parent class'methods and instance data
C) access a child class'constructor(s)
D) access a child class'methods and instance data
E) none of the above

Answer: E

2 True/False Questions

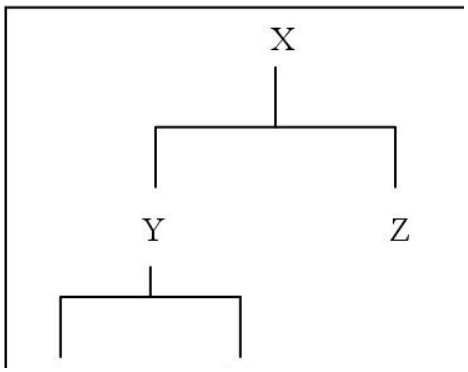
22) A derived class has access to all of the methods of the parent class, but only the protected or public instance data of the parent class.

Answer: FALSE

23) If class `AParentClass` has a protected instance data `x`, and `AChildClass` is a derived class of `AParentClass`, then `AChildClass` can access `x` but can not redefine `x` to be a different type.

Answer: FALSE

Consider the following class hierarchy and answer the questions #24 ~ 26.



24) A is a derived class of X.

Answer: FALSE

Explanation: While A is a descendant of class X, it is not a derived class. Instead, X is a derived class of Y and Y is a derived class of X. A ultimately inherits items from X as long as those items were not overridden by Y.

25) Y is a derived class of Z.

Answer: FALSE

26) If A, B, Y and Z all contain the same instance data d1, then d1 should be declared in X and inherited into Y, Z, A and B.

Answer: TRUE

27) If classes `C1` and `C2` both implement an interface `Cint`, which has a method `whichIsIt`, and if `C1 c = new C1( );` is performed at one point of the program, then a later instruction `c.whichIsIt( );` will invoke the `whichIsIt` method defined in `C1`.

Answer: FALSE

For the questions # 28 ~ 29 below, assume that Poodle is a derived class of Dog and that Dog d = new Dog(...) and Poodle p = new Poodle(...) where the ... are the necessary parameters for the two classes.

28) The assignment statement `d = p;` is legal even though `d` is not a `Poodle`.

Answer: TRUE

29) The assignment statement `p = d;` is legal even though `p` is not a `Dog`.

Answer: FALSE

30) The reserved word, `extends`, is used to denote a has-a relationship.

Answer: FALSE

31) The protected visibility modifier provides the best possible encapsulation that permits inheritance.

Answer: TRUE

32) You may use the `super` reserved word to access a parent class' private members.

Answer: FALSE

33) Java doesn't support multiple inheritance; but it does support the implementation of multiple Interfaces.

Answer: TRUE

34) Although classes can be inherited from one-another, even Abstract classes, interfaces cannot be inherited.

Answer: FALSE

3 Free-Form Questions

35) Assume a class `Triangle` has been defined. It has three instance data, `Point p1, p2, p3`. The class also has a constructor that receives the 3 Points and initializes `p1`, `p2`, and `p3`, a `toString` method that outputs the 3 Points, and a `computeSurfaceArea` method that calculates the surface area of the `Triangle` (as a `double` value). We want to extend this class to represent a 3-dimensional `Pyramid`, which consists of 4 Points. Since the `Pyramid` will consist of in essence, 4 triangles, `computeSurfaceArea` for the `Pyramid` will compute the surface area of the `Triangle` made up of 3 of those 4 Points. Write the `Pyramid` class to handle the difference(s) between a `Pyramid` and the previously defined `Triangle`. Assume the instance data of `Triangle` are protected and all methods are public. Also assume that the class Point has a `toString` method.

Answer:
```
public class Pyramid extends Triangle
{
      protected Point p4;

      public Pyramid(Point a1, Point a2, Point a3, Point a4)
      {
            super(a1, a2, a3);
            p4 = a4;
      }

      public String toString( )
      {
            super.toString( ) + p4.toString( );
      }

      public double computeSurfaceArea(Point a1, Point a2, Point a3, Point a4)
      {
            double total = 0;
            total = super.computeSurfaceArea(a1, a2, a3) +
                    super.computeSurfaceArea(a1, a2, a4) +
                    super.computeSurfaceArea(a1, a3, a4) +
                    super.computeSurfaceArea(a2, a3, a4);

            return total;
      }
}
```

36) Assume the class `Student` implements the `Speaker` interface from the textbook. Recall that this interface includes two abstract methods, `speak( )` and `announce(String str)`. A `Student` contains one instance data, `String classRank`. Write the `Student` class so that it implements `Speaker` as follows. The `speak` method will output "`I am a newbie here`" if the `Student` is a "`Freshman`", "`I hate school`" if the Student is either a "`Sophomore`" or a "`Junior`", or "`I can not wait to graduate`" if the `student` is a "`Senior`". The announce method will output "`I am a Student, here is what I have to say`" followed by the String parameter on a separate line. Finally, the `classRank` is initialized in the constructor. Only implement the constructor and the methods to implement the `Speaker` interface.

Answer:
```
public class Student implements Speaker
{
      private String classRank;

      public Student(String cr)
      {
            classRank = cr;
      }

      public void speak( )
      {
            if (cr.equals("Freshman"))
                  System.out.println("I am a newbie here");
            else if (cr.equals("Sophomore") | | cr.equals("Junior"))
                  System.out.println("I hate school");
            else if (cr.equals("Senior"))
                  System.out.println("I can not wait to graduate");
      }

      public void announce(String str)
      {
            System.out.println("I am a Student, here is what I have to say");
            System.out.println(str);
      }
}
```

Consider the class Car and the subclass SportsCar. Car has instance data of currentSpeed, type, year, cost, model, color and methods of accelerate, brake, getGasMileage, getInsuranceRate and determineSpeed. Use this information to answer the questions below.

37) What instance data and methods might you define for `SportsCar` that are not part of `Car`?

Answer: A sports car may have a racing stripe, which could be stored in a boolean variable, and a maxSpeed, also the number of gears. So the SportsCar class might add instance data of stripe, maxSpeed and numberGears. There might be methods to determine how safe the SportsCar is and whether it could win a race or not against another SportsCar.

38) What methods inherited from `Car` should `SportsCar` override?

Answer: Because a SportsCar will have different acceleration, the accelerate method should be overridden. Also, determineSpeed might change as will getInsuranceRate. The brake method may or may not differ and getGasMileage will probably stay the same.