AP Computer Science A
Semester 1 Practice Final #1 Multiple Choice

Name ____
Date _______________________________ Period ___________

1. What is the value of **product** after the following code segment is executed?

```
int [] factors = {2, 3, 4, 7, 2, 5};
int product = 1;
for (int i = 1; i < factors.length; i +=2)
{
        product *= (factors[i] % factors[i-1]);
}
```

(A) **2**
(B) **3**
(C) **4**
(D) **5**
(E) **9**

2. Consider the following method:

```
public void mystery (int a, int b)
{
        System.out.print(a + "  ");
        if (a <= b)
        {
                mystery (a + 5, b - 1);
        }
}
```

What is the output when **mystery(0,16)** is called?

(A) **0**
(B) **0 5**
(C) **0 5 10**
(D) **0 5 10 15**
(E) **0 5 10 15 20**

3. Which of the following statements about **interfaces** is true?

I. An **interface** contains only public **abstract** methods and public static final fields.

II. If a class implements an **interface** and then fails to implement any methods in that **interface**, then the class must be declared **abstract .**

III. While a class may implement just one **interface**, it may extend more than one class.

(A) **I** only
(B) **I** and **II** only
(C) **I** and **III** only
(D) **II** and **III** only
(E) **I**, **II**, and **III**

4. Which of the following declarations will cause an **error**?

I. ArrayList<String> stringList = new ArrayList<String>();

II. ArrayList<int> intList = new ArrayList<int>();

III. ArrayList<Comparable> compList = new ArrayList<Comparable>();

(A) **I** only
(B) **II** only
(C) **III** only
(D) **I** and **III** only
(E) **II** and **III** only

5. Consider the following code fragment. You may assume that **wordList** has been declared as **ArrayList<String>**.

```
for (String s : wordList)
{
        if (s.length() < 4)
        {
                System.out.println("This word is short.");
        }
}
```

What is the maximum number of times that **"This word is short."** can be printed?

(A) **3**
(B) **4**
(C) **wordList.size()**
(D) **wordList.size() - 1**
(E) **s.length()**

6. What is the size of a **double** variable in Java?

(A) 2 bytes
(B) 4 bytes
(C) 8 bytes
(D) It depends on the compiler setting
(E) It depends on the operating system

7. If **Crossword** extends **Puzzle** and implements **Solvable**, and

```
Puzzle p = new Puzzle();
Crossword cw = new Crossword(10, 20);
```

are declared, which of the following statements will cause a syntax **error**?

(A) **p = cw;**
(B) **cw = new Puzzle();**
(C) **p = new Crossword(10, 20);**
(D) **Solvable x = new Crossword(10, 20);**
(E) All of the above will compile with no errors

8. The expression **!((x <= y) && (y > 5))** is equivalent to which of the following?

(A) **(x <= y) && (y > 5)**
(B) **(x <= y) || (y > 5)**
(C) **(x >= y) || (y < 5)**
(D) **(x > y) || (y <= 5)**
(E) **(x > y) && (y <= 5)**

9. What will be printed by the following code fragment?

```
for (int i = 5; i > 0; i--)
{
        for (int k = 1; k <= i; k++)
        {
                System.out.print(k * k + "  ");
        }
        System.out.println();
}
```

(A)
```
1
1  4
1  4  9
1  4  9  16
1  4  9  16  25
```

(B)
```
1  4  9  16  25
1  4  9  16
1  4  9
1  4
1
```

(C)
```
25  16  9  4  1
25  16  9  4
25  16  9
25  16
25
```

(D)
```
25
25  16
25  16  9
25  16  9  4
25  16  9  4  1
```

(E)
```
1  4  9  16  25
1  4  9  16  25
1  4  9  16  25
1  4  9  16  25
1  4  9  16  25
```

10. The following method is intended to remove from **List<Integer> list** all elements whose value is less than zero:

```
public void removeNegatives (List<Integer> list)
{
    int i = 0, n = list.size();

    while (i < n)
    {
        if (list.get(i) < 0)
        {
            list.remove(i);
            n--;
        }
        i++;
    }
}
```

For which list of **Integer** values does this method work as intended?

(A) Only an empty list
(B) All lists that do not contain negative values in consecutive positions
(C) All lists where all the negative values occur before all the positive values
(D) All lists where all the positive values occur before all the negative values
(E) All lists

11. Determine what is printed from the following code fragment:

```
String s = new String("ONION");
System.out.println(s.substring(1, 5).substring(1, 4).substring(0, 3));
```

(A) **I**
(B) **IO**
(C) **ION**
(D) **ONI**
(E) **NION**

12. Refer to the **doSomething** method below:

```
public static void doSomething (ArrayList<SomeType> list,  int i,  int j)
{
    SomeType temp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, temp);
}
```

Which best describes the post condition for **doSomething**?

(A) Removes from **list** the objects indexed at **i** and **j**.
(B) Replaces in **list** the object indexed at **i** with the object indexed at **j**.
(C) Replaces in **list** the object indexed at **j** with the object indexed at **i**.
(D) Replaces in **list** the objects indexed at **i** and **j** with temp.
(E) Interchanges in **list** the objects indexed at **i** and **j**.

Questions **13** and **14** refer to the following class, **SortOf**:

```
public class SortOf
{
        public static void sort (String [] items)
        {
                int n = items.length;
                while (n > 1)
                {
                        sortHelper(items,n - 1);
                        n--;
                }
        }

        private static void sortHelper (String [] items, int last)
        {
                int m = last;
                for(int k = 0; k < last; k++)
                {
                        if(items[k].compareTo(items[m]) > 0)
                        {
                                m = k;
                        }
                }
                String temp = items[m];
                items[m] = items[last];
                items[last] = temp;
        }
}
```

13. The sorting algorithm implemented in the **sort** method can best be described as:

(A) a **Selection** Sort variation
(B) an **Insertion** Sort variation
(C) a **Bubble** Sort variation
(D) a **Merge** Sort variation
(E) Incorrect implementation of a sorting algorithm

14. Suppose **names** is an array of **String** objects:

```
String [] names = {"Dan", "Alice", "Claire", "Evan", "Boris"};
```

If **SortOf.sort(names)** is running, what is the order of the values in **names** after two complete iterations through the **while** loop in the **sort** method?

(A) **"Boris", "Alice", "Claire", "Dan", "Evan"**
(B) **"Alice", "Claire", "Boris", "Dan", "Evan"**
(C) **"Alice", "Boris", "Claire", "Evan", "Dan"**
(D) **"Alice", "Claire", "Dan", "Evan", "Boris"**
(E) None of the above

15. Consider the following class definitions:

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

public class Holland extends Country
{
        public String toString()
        {
                return "Holland";
        }
}
```

What is the output of the following code fragment?

```
        Country country1 = new Country();
        Country country2 = new Holland();
        System.out.print(country1 + "  " + country2 + "  ");
        country1 = country2;
        System.out.print(country1);
```

(A) **Country Country Country**
(B) **Country Country Holland**
(C) **Country Holland Country**
(D) **Country Holland Holland**
(E) **Holland Holland Holland**

16. Consider the following methods:

```
        public int fun1 (int n)                 |           public int fun2 (int n)
        {                                       |           {
                int product = 1;                |                   int product = 1;
                int k;                          |                   int k = 2;
                for (k = 2; k <= n; k++)        |                   while (k <= n)
                {                               |                   {
                        product *= k;           |                           product *= k;
                }                               |                           k++;
                return product;                 |                   }
        }                                       |                   return product;
                                                |           }
```

For which integer values of **n** do **fun1(n)** and **fun2(n)** return the same result?

(A) Only **n > 1**
(B) Only **n < 1**
(C) Only **n == 1**
(D) Only **n >= 1**
(E) Any integer **n**

17. Consider the following class:

```
public class ClassList
{
        //  contains the list of student names in a class.
        private ArrayList<String> studentnames;

        //  constructors and other methods and variables not shown.

        public ArrayList<String> getnames ( )
        {
                return studentnames;
        }
}
```

If **ClassList myClass** is declared and initialized in some other client class, which of the following correctly assigns to name the name of the first student in the **myClass** list?

I. String name = myClass.studentnames[0];
II. String name = myClass.studentnames.get(0);
III. String name = myClass.getnames().get(0);

(A) **I** only
(B) **II** only
(C) **III** only
(D) **I** and **II** only
(E) **II** and **III** only

18. Consider the following method:

```
public int twonumbers (int k, int n)
{
        if (n == k)
        {
                return k;
        }
        else if (n > k)
        {
                return twonumbers(k, n - k);
        }
        else
        {
                return twonumbers(k - n, n);
        }

}
```

What value is returned by the call **twonumbers(15, 21)** ?

(A) **21**
(B) **15**
(C) **5**
(D) **3**
(E) **1**

For questions **19** and **20**, consider the following interfaces and classes:

```
public interface InterfaceA
{
        public void methodA ( );
}

public interface InterfaceB extends InterfaceA
{
        public void methodB ( );
}

public class ClassA implements InterfaceA
{
        public void methodA ( )
        {
        }

        public void methodB ( )
        {
        }
}

public class ClassB extends ClassA implements InterfaceB
{
        public ClassB ( )
        {
        }

         //  other methods, constructors, and data members not shown . . .
}
```

19. What is the minimum requirement for method definitions in **ClassB** for it to compile with no errors?

(A) No further method definitions are required
(B) **methodA**
(C) **methodB**
(D) **methodA** and **methodB**
(E) **methodA**, **methodB**, and **toString**

20. Which of the following statements would cause a syntax **error**?

(A) **InterfaceA obj1 = new ClassA();**
(B) **InterfaceB obj2 = new ClassA();**
(C) **InterfaceA obj3 = new ClassB();**
(D) **InterfaceB obj4 = new ClassB();**
(E) **ClassA obj5 = new ClassB();**

21. Which of the following does **NOT** print out "**2/3**"?

(A) **System.out.println("2/3");**
(B) **System.out.println("2" + "/" + "3");**
(C) **System.out.println(2/3);**
(D) **System.out.println(2 + "/" + 3);**
(E) **System.out.println((int)2 + "/" + (int)3);**

22. What is the **scope** of a variable?

(A) The range of values that the variable can assume
(B) The number of bytes the variable occupies in memory
(C) The period of time from the point the variable is initialized in the program to the point when it is destroyed
(D) The place(s) in the code where the variable exists and is accessible
(E) The name of the variable

23. What is a client class of class **X** ?

(A) Any subclass of **X**
(B) A class whose objects serve as **X**'s fields
(C) Any class above or below **X** in the inheritance line
(D) Any class in the same folder as **X**
(E) Any class that invokes **X**'s constructors and calls its methods.

24. Consider the following class:

```
public class TestPow2
{
        private static int [] powersOfTwo = {1, 2, 4, 8, 16, 32};
        private int num;

        // other methods, data members, and constructors not shown . . .
}
```

Which of the following methods inside the **TestPow2** class will compile with no errors?

```
I.      public int pow2 ( )
        {
                return powersOfTwo[num];
        }

II.     public static int pow2 (int x)
        {
                return powersOfTwo[x];
        }

III.    public static int pow2 ( )
        {
                return powersOfTwo[num];
        }
```

(A) **I** only (B) **II** only (C) **I** and **II** only (D) **II** and **III** only (E) **I**, **II**, and **III**

Refer to the following class for Questions **25** and **26**.

```
public class Tester
{
        private int [ ] testArray = {3, 4, 5};

        public void increment (int n)
        {
                n++;
        }

        public void firstTestMethod ( )
        {
                for(int i = 0; i < testArray.length; i++)
                {
                        increment(testArray[i]);
                        System.out.print(testArray[i] + " ");
                }
        }

        public void secondTestMethod ( )
        {
                for (int element : testArray)
                {
                        increment (element);
                        System.out.print(element + " ");
                }
        }
}
```

25. What output will be produced by calling **firstTestMethod** for a **Tester** object (assuming that **testArray** contains **3 4 5**)?
(A) **3 4 5**
(B) **4 5 6**
(C) **5 6 7**
(D) **0 0 0**
(E) No output will be produced. An **ArrayIndexOutOfBoundsException** will be thrown.

26. What output will be produced by invoking **secondTestMethod** for a **Tester** object (assuming that **testArray** contains **3 4 5**)?
(A) **3 4 5**
(B) **4 5 6**
(C) **5 6 7**
(D) **0 0 0**
(E) No output will be produced. An **ArrayIndexOutOfBoundsException** will be thrown.

27. What happens if you forget to initialize a field that is an object, and start calling its methods?
(A) A syntax error is reported
(B) A run-time "null reference exception" error is reported
(C) A new object is created with the default values for its fields
(D) A new object is created with unpredictable values for its fields
(E) Mr. DeRuiter will tap you on the shoulder and say "bad programmer!" (this is not the correct answer . . .)

28. Consider the following code segment.

```
double a = 1.1;
double b = 1.2;

if ((a + b) * (a - b) != (a * a) - (b * b))
{
        System.out.println("Mathematical error!");
}
```

Which of the following best describes why the phrase **"Mathematical error!"** would be printed?
(Remember that mathematically $(a+b)*(a-b) = a^2 - b^2$.)

(A) Precedence rules make the **if** condition true.
(B) Associativity rules make the **if** condition true.
(C) Overflow makes the **if** condition true.
(D) A compiler bug or hardware error has occurred.
(E) Roundoff error make the **if** condition true.

29. Which of the following is a good stylistic rule for naming fields?

(A) Start with a lower case letter.
(B) Use a single lower case letter.
(C) Start with an upper case letter.
(D) Use all caps.
(E) Use the names of friends, where possible, like Johnny, Larry, Suzy, and Parul.

30. Consider the following code segment:

```
        int x = 0, y = 3;
        String op = new String("/");

        if (op.equals("/") && (x != 0) && (y/x > 2))
        {
                System.out.println("Okay");
        }
        else
        {
                System.out.println("Failed");
        }
```

Which of the following statements about this code is true?

(A) There will be an error at compile time, because **String** and **int** variables are intermixed in the same condition.
(B) There will be an error at run time, due to division by zero.
(C) The code will compile and execute without error. The output will be **OK .**
(D) The code will compile and execute without error. The output will be **Failed .**
(E) The code will compile and execute without error. There will be no output (nothing printed).

31. Consider the following classes:

```
public class Secret
{
        private int coolnum;

        public Secret (int c)
        {
                coolnum = c;
        }

        public int getcoolnum ( )
        {
                return coolnum;
        }

        public String getletters ( )
        {
                return "Secret";
        }

        public String getMessage ( )
        {
                return getletters() + " - " + getcoolnum();
        }
}
```

```
public class Code extends Secret
{
          public Code (int c)
          {
                    super(c + 1);
          }

          public int getcoolnum ( )
          {
                    return super.getcoolnum() + 1;
          }

          public String getletters ( )
          {
                    return "Code";
          }
}
```

What is the output of the following code fragment?

```
          Secret scramble = new Code(0);
          System.out.println(scramble.getMessage());
```

(A) **Secret - 0**
(B) **Secret - 1**
(C) **Secret - 2**
(D) **Code - 1**
(E) **Code - 2**

32. Consider the following code segment

```
          int [] array = {2, 4, 6, 8, 10, 12};

          for (int k = 2; k < array.length - 1; k++)
          {
                    array[k] = array[k + 1];
          }
```

Which of the following represents the contents of **array** as a result of executing the code segment?

(A) **2, 4, 6, 8, 10, 12**
(B) **2, 6, 6, 8, 10, 12**
(C) **2, 4, 8, 8, 10, 12**
(D) **2, 4, 8, 10, 12, 12**
(E) **2, 6, 8, 10, 12, 12**

33. Consider the following class definitions.

```
public class ClassTwo extends ClassOne
{
        public void methodTwo ( )
        {
        }
}

public class ClassOne
{
        public void methodOne ( )
        {
        }

        public void methodTwo ( )
        {
        }
}
```

Now, consider the following declarations in a client class. You may assume that **ClassOne** and **ClassTwo** have default constructors.

```
        ClassOne c1 = new ClassOne();
        ClassOne c2 = new ClassTwo();
```

Which of the following method calls will cause an error?

I. **c1.methodTwo();**
II. **c2.methodTwo();**
III. **c2.methodOne();**

(A) None of these
(B) **I** only
(C) **II** only
(D) **III** only
(E) **II** and **III** only

34. Consider the following declarations.

```
        public interface Comparable
        {
                public int compareTo(Object other);
        }

        public class SomeClass implements Comparable
        {
                // ... other methods not shown
        }
```

Which of the following method signatures of **compareTo** will satisfy the **Comparable** interface requirement?

I. **public int compareTo(Object other)**
II. **public int compareTo(SomeClass other)**
III. **public boolean compareTo(Object other)**

(A) **I** only
(B) **II** only
(C) **III** only
(D) **I** and **II** only
(E) **I**, **II**, and **III**

35. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best design?

(A) Use one class, **Car**, which has three data fields: **int numDoors**, **boolean hasAir**, and **double milesPerGallon**.
(B) Use four unrelated classes: **Car**, **Doors**, **AirConditioning**, and **MilesPerGallon**.
(C) Use a class **Car** which has three subclasses: **Doors**, **AirConditioning**, and **MilesPerGallon**.
(D) Use a class **Car**, which has a subclass **Doors**, with a subclass **AirConditioning**, with a subclass **MilesPerGallon**.
(E) Use three classes: **Doors**, **AirConditioning**, and **MilesPerGallon**, each with a subclass **Car**.

36. Which of the following statements about interfaces and abstract classes is **TRUE**?

(A) An abstract class cannot extend another abstract class.
(B) If an abstract class has no implemented constructors or methods, it is better to make it an interface.
(C) An abstract class cannot implement an interface.
(D) You can declare an array of objects of an abstract class type, but not of an interface type.
(E) A method can take a parameter of an interface type, but not of an abstract class type.

37. What are "overloaded" methods?

(A) Methods in a derived class that redefine a method from the base class
(B) Methods of a class that create objects of the same class
(C) Methods of the same class that have the same name but different numbers or types of parameters
(D) Methods of different classes that have the same name
(E) Methods that are taking too many final exams

38. Suppose the **isPrime** method is defined:

```
// returns true if p is a prime number, false otherwise
// precondition: p >= 2
public static boolean isPrime (int p)
{
        // code not shown
}
```

Given:

```
int n = 101;
int sum = 0;
```

Which of the following code segments correctly computes the sum of all prime numbers from 2 to 101?

(A)
```
while (n != 2)
{
        n--;
        if (isPrime(n))
        {
                sum += n;
        }
}
```

(B)
```
while (n >= 2)
{
        n--;
        if (isPrime(n))
        {
                sum += n;
        }
}
```

(C)
```
while (n != 2)
{
        if (isPrime(n))
        {
                sum += n;
        }
        n--;
}
```

(D)
```
while (n >= 2)
{
        if (isPrime(n))
        {
                sum += n;
        }
        n--;
}
```

(E)
```
while (n >= 2 && isPrime(n))
{
        sum += n;
        n--;
}
```

39. The number of comparisons in **Bubble** Sort in an array of n elements is roughly proportional to

(A) n
(B) n log n
(C) $(\log n)^2$
(D) n^2
(E) n^3

40. Consider the following statements about the Java programming language:

I. Parameters of primitive data types are always passed **“by value”**.
II. Objects are always passed to methods and constructors as **copies of references**.
III. A method can also have a return type of some class. That means the method returns a **reference** to an object of that class.

Which of the following is true?

(A) **I** only
(B) **I** and **II** only
(C) **II** and **III** only
(D) **I** and **III** only
(E) **I**, **II**, and **III**

41. As a general "rule of thumb" (a general rule), it is a good idea to make

(A) all class instance variables private.
(B) all class instance variables public.
(C) class variables a mix of public and private, according to the purpose of the variable.
(D) class variables a mix of public, protected, and private, according to the purpose of the variable.
(E) class variables a mix of protected and private, according to the purpose of the variable.

42. Given the declaration
Object obj = new String("Finals Week!");
Which of the following expressions will compile with no errors?

I.
```
System.out.println(obj.substring(3,9));
```

II.
```
if (obj.equals("Finals Week!"))
{
        System.out.println("Great!");
}
```

III.
```
if (obj.compareTo("Finals Week!") == 0)
{
        System.out.println("Terrific!");
}
```

(A) **I** only (B) **II** only (C) **I** and **II** only (D) **II** and **III** only (E) **I**, **II**, and **III**

43. The class **CourseList** provides methods that allow you to represent and manipulate a list of high school courses, but you are not concerned with how these operations work or how the list is stored in memory. You only know how to initialize and use **CourseList** objects and have no direct access to the implementation of the **CourseList** class or its private data fields. This is an example of
(A) encapsulation
(B) overriding
(C) inheritance
(D) polymorphism
(E) method overloading

44. What is the output of the following code segment?

```
String mv1 = "Can't wait ";
String mv2 = mv1;
mv2 += "for vacation! ";
System.out.println(mv1 + mv2);
```

(A) **"Can't wait Can't wait for vacation!"**
(B) **"Can't wait for vacation! Can't wait for vacation!"**
(C) **"Can't wait for vacation!"**
(D) **"Can't wait for vacation! for vacation!"**
(E) **"Can't wait for school to start again!"**

45. What is the result when the following code segment is compiled and executed?

```
int m = 4, n = 5;
double d = Math.sqrt ((m + n) / 2);
System.out.println (d);
```

(A) the following syntax error occurs: "**sqrt(double) in java.lang.Math cannot be applied to int**"
(B) **1.5** is printed
(C) **2.0** is printed
(D) **2.1213203435596424** is printed
(E) a **ClassCastException** is thrown

46. Consider the following code segment:

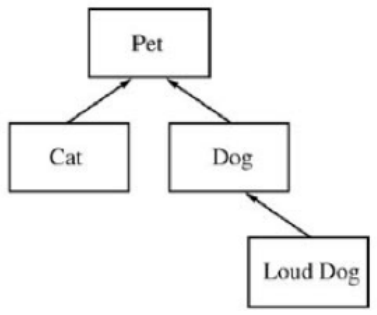
```
while (x > y)
{
        x--;
        y++;
}
System.out.println(x - y);
```

Assume that **x** and **y** are **int** variables and their values satisfy the conditions **0 <= x <= 2** and **0 <= y <= 2** before the start of the while loop. Which of the following describes the set of all possible outputs?

(A) **0**
(B) **-1, 1**
(C) **-1, -2**
(D) **0, -1, -2**
(E) **0, -1, 1, -2, 2**

FREE RESPONSE

1. Consider the hierarchy of classes shown in the following diagram.



Note that a **Cat** "is-a" **Pet**, a **Dog** "is-a" **Pet**, and a **LoudDog** "is-a" **Dog**. The class **Pet** is shown in the following declaration.

```
public class Pet
{
        private String myName;

        public Pet (String name)
        {  myName = name;  }

        public String getName()
        {  return myName;  }

        public String speak()
        {  return "ah-OOOOO";  }
}
```

The subclass **Dog** has the partial class declaration shown below.

```
public class Dog extends Pet

{
        public Dog (String name)
        {  /* implementation not shown  */  }

        public String speak ()
        {  /* implementation not shown  */  }
}
```

(a) Given the class hierarchy shown above, write a complete class declaration for the class **Cat**, including implementations of its constructor and method(s). The **Cat** method **speak** returns “meow” when it is invoked. Note that the method **speak** will need to be overridden in **Cat**.

(b) Assume that class **Dog** has been declared as shown at the beginning of the question. If the String *dog-sound* is returned by the **Dog** method **speak**, then the **LoudDog** method **speak** returns a String containing *dog-sound* repeated two times. (note that we do not know the String for *dog-sound*)

Given the class hierarchy shown previously, write a complete class declaration for the class **LoudDog**, including implementations of its constructors and method(s).

2. Consider a class, **NoteKeeper**, that is designed to store and manipulate a list of short notes. Here are some typical notes:

```
pick up drycleaning
special dog chow
dog registration
dentist Monday
study for APCS quiz
```

The incomplete class declaration is shown below:

```
public class NoteKeeper
{
        private ArrayList<String> noteList;

        // Postcondition:  Prints all notes in noteList, one per line.
        //                 Notes are numbered  1, 2, 3, …
        //                 Each number is followed by a period and a space.
        public void printNotes ( )
        {
                /*  to be implemented in part (a)  */
        }

        // Postcondition:  All notes with specified word have been removed from noteList, leaving the order of the
        //                 remaining notes unchanged.  If none of the notes in noteList contains word, the list remains
        //                 unchanged.
        public void removeNotes(String word)
        {
                /*  to be implemented in part (b)  */
        }

        // Constructor and other methods not shown . . .
}
```

(a) Write the **NoteKeeper** method **printNotes**. The **printNotes** method prints all of the notes in **noteList**, one per line, and numbers the notes, starting at 1. The output should look like this:

```
1.  pick up drycleaning
2.  special dog chow
3.  dog registration
4.  dentist Monday
5.  study for APCS quiz
```

Complete method **printNotes** below.

```
// Postcondition:  Prints all notes in noteList, one per line.
//                 Notes are numbered  1, 2, 3, …
//                 Each number is followed by a period and a space.
public void printNotes ( )
```

(b) Write the **NoteKeeper** method **removeNotes**. Method **removeNotes** removes all notes from **noteList** that contain the word specified by the parameter. The ordering of the remaining notes should be left unchanged. For example, suppose that a **NoteKeeper** variable, **notes**, has a **noteList** containing

[pick up drycleaning, special dog chow, dog registration, dentist Monday, study for APCS quiz]

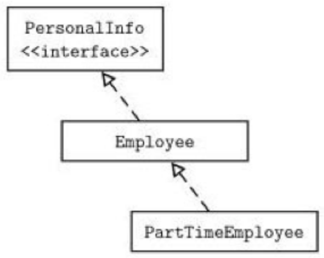
The method **notes.removeNotes("dog")** should modify the **noteList** of **notes** to be

[pick up drycleaning, dentist Monday, study for APCS quiz]

Complete method **removeNotes** below.

```
// Postcondition:  All notes with specified word have been removed from noteList, leaving the order of the
//                 remaining notes unchanged.  If none of the notes in noteList contains word, the list remains
//                 unchanged.
public void removeNotes(String word)
```

3. Consider the class hierarchy diagram shown to the right.



PersonalInfo is an interface that is implemented by **Employee**. A **PartTimeEmployee** is-a **Employee**. Here is the declaration for **PersonalInfo**.

```
public interface PersonalInfo
{
        String getName();
        String getCity();
        boolean getCitizenStatus();
}
```

(a) Given the class hierarchy diagram shown above, write a complete class declaration for the class **Employee**, including implementation of methods and a constructor with parameters. An **Employee**, in addition to implementing **PersonalInfo**, has a data field to store annual salary, and an accessor method that returns the annual salary of the **Employee**. Write the **Employee** class below, or on the multiple choice answer sheet.

(b) Write a complete class declaration for the **PartTimeEmployee** class, given the class hierarchy shown above. A **PartTimeEmployee** has two additional data fields:

- A floating point number that indicates the fraction that the part-time employee works (for example, 0.5, 0.8, etc.)
- A boolean field that stores whether the employee is a union member or not.

There are three additional methods:

- An accessor that returns the floating point number fraction that the **PartTimeEmployee** works.
- An accessor that returns a value indicating whether the **PartTimeEmployee** is a union member or not.
- A mutator that switches the status of that employee's union membership. If the employee's salary is greater than $15,000, then employee is not a union member. Otherwise, the employee is a union member.

Write the **PartTimeEmployee** class below, or on the multiple choice answer sheet.