1. Consider the following code segment.

```
String jolly = new String("Ho");
jolly.toUpperCase();
System.out.println(jolly + jolly.toLowerCase());
```

What is printed as a result of running the code segment?

(A) Hoho
(B) HOho
(C) HoHo
(D) hoHO
(E) falalalala-lalalala

2. Consider the following classes.

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

public class B extends A
{

}

public class C extends B
{
        public String toString ( )
        {
                return super.toString() + "C";
        }

        public static void main (String [] args)
        {

        }
}
```

What is printed as a result of running the following code segment?

```
A a = new C();
System.out.println(a);          //  note that this will invoke a toString() method
```

(A) A
(B) C
(C) AC
(D) ACDC
(E) ABCD

3. Consider the following code segment.

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(2);
list.add(1);
list.add(0);
int n = list.size();
for (int i = 0; i < n; i++)
{
        int value = list.get(i);
        if (value > 0)
                list.add(0,value);
}
System.out.println(list);
```

What is printed as a result of running the code segment?

(A) [2, 1, 0]
(B) [2, 1, 0, 1, 2]
(C) [2, 1, 0, 2, 1]
(D) [2, 1, 2, 1, 0]
(E) [2, 2, 2, 2, 1, 0]

4. What values are stored in the array **arr** after the following code is executed?

```
int [] arr = {1, 2, 3, 4, 5};
int num = 0;
for (int i = 0; i < arr.length; i++)
{
        num += arr[i];
        arr[i] = num;
}
```

(A) 0, 0, 0, 0, 0
(B) 1, 2, 3, 4, 5
(C) 0, 1, 3, 6, 10
(D) 1, 3, 6, 10, 15
(E) 1, 4, 10, 20, 35

5. Suppose **x** is an **int** variable that holds a positive integer. Consider the following five expressions:

a. **x % 100 / 10**
b. **x / 10 % 10**
c. **(x - x % 100) / 10**
d. **x / 10 - x / 100 * 10**
e. **(x - x / 100 * 100) / 10**

Which one of them produces a value different from the other four for some values of x?

(A) a.
(B) b.
(C) c.
(D) d.
(E) e.

6. The relationship between a child (sub) class and a parent (super) class is referred to as a _____ relationship.

(A) has-a
(B) was-a
(C) instance-of
(D) is-a
(E) whatever-a

7. Suppose an interface **Polygon** specifies the **getArea()** method. Two classes, **Triangle** and **Hexagon**, implement **Polygon**. Which Java feature makes it possible for the following code segment to print the correct values for the area of a triangle and a hexagon?

```
Polygon [] polygons = new Polygon[2];
polygons[0] = new Triangle(3,4,5);
polygons[1] = new Hexagon(3,3,3,3,3,3);
System.out.println("Triangle: " + polygons[0].getArea());
System.out.println("Hexagon: " + polygons[1].getArea());
```

(A) abstraction
(B) polymorphism
(C) encapsulation
(D) platform independence
(E) method overloading

8. Consider the following method:

```
// precondition:  x >= 0
public void mystery (int x)
{
        if ((x / 10) != 0)
                mystery(x / 10);
        System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call **mystery(123456)** ?

(A) 16
(B) 56
(C) 123456
(D) 654321
(E) Many digits are printed due to infinite recursion.

9. 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

10. Which of the following statements is equivalent to

```
if (a == 0 || b*b - 4*a*c <= 0)
        return false;
else
        return true;
```

(A) `return a == 0 || b*b - 4*a*c <= 0;`
(B) `return a == 0 && b*b - 4*a*c <= 0;`
(C) `return a != 0 || b*b - 4*a*c > 0;`
(D) `return a != 0 && b*b - 4*a*c > 0;`
(E) `return true;`

11. Consider the following class:

```
public class IQScore
{
        private int score;

        public IQScore (int s)
        {
                score = s;
        }

        public void increase (int add)
        {
                score = score + add;
        }

        public int getScore ( )
        {
                return score;
        }

        public static void main (String [] args)
        {
                IQScore winnie = new IQScore(100);
                IQScore sally = new IQScore(100);
                IQScore harry = sally;

                harry.increase(20);

                System.out.println(winnie.getScore() + "  " + sally.getScore() + "  " + harry.getScore());
        }
}
```

What is printed when the class is compiled and run?

(A) **120 100 120**
(B) **100 120 120**
(C) **100 100 120**
(D) **120 120 100**
(E) **120 120 120**

12. Consider the following method:

```
public int solution (int limit)
{
        int count = 0;

        for (int outer = 1; outer <= limit; outer++)
        {
                for (int inner = outer; inner <= limit; inner++)
                {
                        count++;
                }
        }

        return count;
}
```

What value is returned as a result of the call **solution(7)** ?

(A) 21
(B) 28
(C) 42
(D) 49
(E) 343

13. What is an **abstract** class?

(A) A class that can be hard to interpret, like the modern art of Picasso
(B) A class in which all fields and methods are static
(C) A class where some methods are abstract
(D) A class where all methods are abstract
(E) Any class derived from the **Object** class

14. Which of the following statements is true?

(A) A static variable cannot be initialized in a constructor.
(B) A static variable must be declared final.
(C) A static method can't access an instance variable.
(D) An instance variable can't be declared final.
(E) Only a static method can access a static variable.

For questions 15 – 17, use the following partial class definitions:

```
public class Grandpa
{
        private int bankbalance;
        protected String pocketwatch;
        public String candy;
        …
}

public class Dad extends Grandpa
{
        private int walletmoney;
        protected int changedish;
        …
}

public class Daughter extends Dad
{
        private String facebookpage;
        …
}
```

15. Which of the following lists of instance data are accessible in class **Daughter**?

(A) **bankbalance, pocketwatch, candy, walletmoney, changedish, facebookpage**
(B) **pocketwatch, candy, changedish, facebookpage**
(C) **candy, facebookpage**
(D) **walletmoney, changedish, facebookpage**
(E) **facebookpage**

16. Which of the following lists of instance data are accessible in class **Dad**?

(A) **bankbalance, pocketwatch, candy, walletmoney, changedish, facebookpage**
(B) **pocketwatch, candy, walletmoney, changedish, facebookpage**
(C) **pocketwatch, candy, walletmoney, changedish**
(D) **candy, walletmoney, changedish, facebookpage**
(E) **candy, walletmoney, changedish**

17. Which of the following is true regarding the use of instance data **changedish** of class **Dad**?

(A) it is accessible in **Grandpa**, **Dad**, and **Daughter**
(B) it is accessible in **Dad** and **Daughter**
(C) it is accessible in **Grandpa** and **Dad**
(D) it is accessible in **Dad** only
(E) it is accessible in **Daughter** only

18. Assume that an array of **double** values has been declared as follows and has been initialized.

```
double [] arr = new double[8];
```

Which of the following code segments correctly interchanges the value of **arr[0]** and **arr[4]**?

(A)
```
arr[0] = 4.0;
arr[4] = 0.0;
```

(B)
```
arr[0] = arr[4];
arr[4] = arr[0];
```

(C)
```
double value = arr[4];
arr[0] = arr[4];
arr[4] = value;
```

(D)
```
double value = arr[0];
arr[0] = arr[4];
arr[4] = value;
```

(E)
```
double value = arr[4];
arr[4] = arr[0];
arr[0] = arr[4];
```

19. Consider the following declarations:

```
String s1 = "winter vacation";
String s2 = new String("winter vacation");
String s3 = s1;
```

Which expression(s) involving these Strings evaluates to true?

I. **s1.equals(s2)** II. **s1 == s2** III. **s3.equals(s1)**

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

20. Which of the following indicates that a method does not take any parameters?

(A) Empty parentheses in the method's header.
(B) No parentheses in the method's header.
(C) The keyword **void** in the method's header inside parentheses.
(D) The keyword **void** in the method's header preceding the method's name.
(E) The keyword **null** in the method's header preceding the method's name.

21. Assume that the following variable declarations have been made.

```
double d = Math.random();
double r;
```

Which of the following assigns a value to **r** from the uniform distribution over the range **0.5 <= r < 5.5** ?

(A) **r = d + 0.5;**
(B) **r = d + 0.5 * 5.0;**
(C) **r = d + 5.0;**
(D) **r = d * 5.5;**
(E) **r = d * 5.0 + 0.5;**

22. Consider the following method:

```
public void increment (int [] arr, int value)
{
        while (value > 0)
        {
                value--;
        }
        arr[value]++;
}
```

What values are stored in **myints** after the following code is executed?

```
int [] myints = new int[4];             // values initialized to 0
for (int i = 0; i < myints.length; i++)
{
        increment(myints,i);
}
```

(A) 0, 0, 0, 0
(B) 1, 1, 1, 1
(C) 4, 3, 2, 1
(D) 4, 0, 0, 0
(E) 1, 2, 3, 4

23. Why doesn't Java let you create an object of the **Math** class?

(A) Because **Math** has no fields.
(B) Because **Math** has no parent class.
(C) Because all **Math**'s methods and fields are static, so all **Math** objects would be identical.
(D) Because **Math** does not represent a real-world object.
(E) Because **Math**'s coolness will not let it be objectified.

24. Consider the following method, **isSorted**, which is intended to return **true** if an array of integers is sorted in nondecreasing order and to return **false** otherwise.

```
public static boolean isSorted (int [] data)
{
        /*  missing code  */
}
```

Which of the following can be used to replace **/* missing code */** so that **isSorted** will work as intended?

I.
```
for (int k = 0; k < data.length; k++)
{
        if (data[k] > data[k + 1])
                return false;
}
return true;
```

II.
```
for (int k = 1; k < data.length; k++)
{
        if (data[k - 1] > data[k])
                return false;
}
return true;
```

III.
```
for (int k = 0; k < data.length - 1; k++)
{
        if (data[k] > data[k + 1])
                return false;
        else
                return true;
}
return true;
```

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

25. Which of the following best describes the base case(s) in the following recursive method?

```
public int factorial (int n)
{
        int product = 1;
        if (n > 1)
                product = n * factorial(n-1);
        return product;
}
```

(A) The method does not have a base case
(B) **n > 0**
(C) **n > 1**
(D) **n <= 1**
(E) **product == 0**

26. 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.

27. 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!"**

28. Consider the following code segment.

```
int k = 0;
while (k < 10)
{
        System.out.print((k % 3) + "  ");
        if ((k % 3) == 0)
                k = k + 2;
        else
                k++;
}
```

What is printed as a result of executing the code segment?

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

29. Which of the following is **not** a method of **java.util.ArrayList<String>**?

(A) **add (String x);**
(B) **add (int index, String x);**
(C) **remove (String x);**
(D) **insert (int index, String x);**
(E) **set (int index, String x);**

30. Consider the following code segment.

```
int a = 24;
int b = 30;
while (b != 0)
{
        int r = a % b;
        a = b;
        b = r;
}
System.out.println(a);
```

What is printed as a result of executing the code segment?

(A) **0**
(B) **6**
(C) **12**
(D) **24**
(E) **30**

31. What is the purpose of indentation in programs?

(A) To make the code more readable.
(B) To mark compound statements for the compiler.
(C) To mark nested loops for the compiler.
(D) To mark methods for the compiler.
(E) So that DeRuiter does not complain and take a point off when he grades your program (this is not the correct answer!).

32. Consider the following method.

```
public String modifyString (String input)
{
        String output = new String("");

        for (int k = 1; k < input.length(); k = k + 2)
        {
                output += input.substring(k, k + 1);
        }
        return output;
}
```

What is returned as a result of the call to **modifyString ("COMPUTER")** ?

(A) **"CMUE"**
(B) **"OPTR"**
(C) **"OMPUTER"**
(D) **"COMPUTE"**
(E) **"COMPUTER"**

33. 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.

34. 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**

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

(A) 2 bytes
(B) 4 bytes
(C) 8 bytes
(D) 16 bytes
(E) It depends on the operating system

36. How many primitive data types are defined in Java?

(A) 3
(B) 4
(C) 5
(D) 8
(E) As many as the programmer defines.

37. All classes in Java are directly or indirectly subclasses of the _____ class.

(A) **Wrapper**
(B) **String**
(C) **Reference**
(D) **this**
(E) **Object**

38. 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

39. What happens if **str.length()** is **8** and you call **str.charAt(8)** ?

(A) **charAt** returns **-1**.
(B) **charAt** returns the last character of **str**.
(C) **charAt** returns the random character that is stored after the **String str**.
(D) **charAt** returns **0**.
(E) **StringIndexOutOfBoundsException**

40. The statement **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

For problems 41 and 42, use the following classes:

```
public abstract class Movie
{
        private String title;

        public Movie (String t)
        {
                title = t;
        }

        public String getTitle ( )
        {
                return title;
        }

        public void setTitle (String t)
        {
                title = t;
        }

        public abstract String getRating();
}

public class MovieWithRating extends Movie
{
        private String rating;

        public MovieWithRating (String t, String r)
        {
                super(t);          // LINE 1
                rating = r;
        }

        public String getRating ( )
        {
                return rating;
        }
}
```

41. Which of the following can replace **LINE 1** in the **MovieWithRating** constructor, so that

```
MovieWithRating mov = new MovieWithRating("Breaking Dawn", "PG-13");
System.out.println(mov.getTitle());
```

will print

```
Breaking Dawn
```

(A) **title = t;**
(B) **setTitle(t);**
(C) **super.setTitle(t);**
(D) both (B) and (C) will work
(E) None of the above will work

42. Suppose we want

```
System.out.println(new MovieWithRating("The Muppets","PG"));
```

to display

```
The Muppets, PG
```

Which of the following will work?

I. Adding the method

```
public String toString ( )
{
        return getTitle() + ", " + getRating();
}
```

to the **Movie** class.

II. Adding the method

```
public String toString ( )
{
        return getTitle() + ", " + getRating();
}
```

to the **MovieWithRating** class.

III. Adding the method

```
public String toString ( )
{
        return getTitle() + ", " + rating;
}
```

to the **MovieWithRating** class.

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

43. Suppose **numbers** is an **ArrayList<Integer>,** and **System.out.println(numbers)** displays

[1, 5, 15, 50, 75, 100]

What is the output when the following code segment is executed?

```
for (int i = 0; i < numbers.size(); i++)
{
        if (numbers.get(i).intValue() % 5 == 0)
        {
                numbers.remove(i);
        }
}
System.out.println(numbers);
```

(A) **[1, 5, 15, 50, 75, 100]**
(B) **[1, 15, 75]**
(C) **[1, 100]**
(D) **[1]**
(E) IndexOutOfBoundsException

44. Consider the following code segment.

```
int sum = 0;
int k = 1;

while (sum < 12 || k < 4)
{
        sum += k;
}

System.out.println(sum);
```

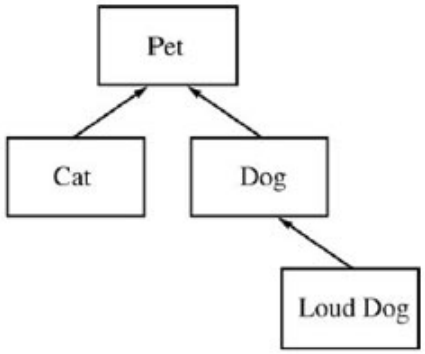
What is printed as a result of executing the code segment?

(A) 6
(B) 10
(C) 12
(D) 15
(E) Nothing is printed due to an infinite loop.

45. Assume that **myList** is an **ArrayList** that has been correctly constructed and populated with objects. Which of the following expressions produces a valid random index for **myList**?

A. **(int)(Math.random() * myList.size()) - 1**
B. **(int)(Math.random() * myList.size())**
C. **(int)(Math.random() * myList.size()) + 1**
D. **(int)(Math.random() * (myList.size() + 1))**
E. **Math.random(myList.size())**

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

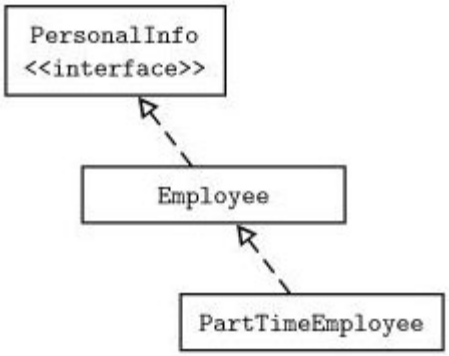
[pick up drycleaning, special dog chow, dog registration, dentist Monday, study for APCS quiz]

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 real 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 real 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, but only if the employee's salary is greater than $15,000. In other words, if his/her salary is greater than $15,000, then the union membership status should be "flipped" from union to nonunion or vice versa.

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