Questions and Exercises for Review

Note. Questions preceded by green numbers apply to the midterm

Questions for final start here

1. Explain the term *data abstraction*.

2. What are the benefits of abstract data types?

3. What is the effect of the <<< operator?

4. What makes the Object Oriented paradigm?

5. Explain the strengths of object orientation.

6. What is abstraction?

7. What are software objects?

8. What are object capabilities?

9. What kind of object capabilities can be defined?

10. What are object properties?

11. What kind of object properties can be defined?

12. How can one get a Java standalone application?

13. What is an object's state?

14. Explain the term *object class*.

15. What is the role of *encapsulation*?

16. Explain the terms *public view* and *private view* as related to a class.

17. Explain *composition* abstraction.

18. Explain how can one get from an ADT to Java code?

19. Explain the steps needed to get from source code to the execution of a Java program.

20. What is a java stand-alone application?

21. What is an applet?

22. What is the role of the class loader?

23. Explain the access modifiers on variables/methods.

24. Give examples of names for variables, methods, classes and constants which obey the naming convention.

25. What are the primitive types in Java?

26. Explain the assignment compatibility for primitive types in Java.

26. What is *type casting*? Give 2-3 relevant examples.

27. What types have wrapper classes and what are the corresponding names of wrapper classes.

28. What is *boxing* and *un-boxing*?

29. What kind of comments can be used in Java programs?

30. What should one use when comparing the contents of objects?

31. Give an example of overflow error in Java.

32. What is the effect of *break with a label*?

33. What is the effect of *continue with a label*?

34. Explain and exemplify the term *constructor* in Java.

35. Explain and exemplify the term *method* in Java.

36. What is the input and output to a method?

37. Provide a template for class code.

38. Why is it useful to overload constructors?

39. What can be an operand to a Java expression?

40. What is taken into account when evaluation an expression?

41. What are the relational operators?

42. Write a relational expression which would evaluate to true if the sum of variables x and y was equal to the value of a variable z.

43. Bracket the following logical expressions to show the order of evaluation of the operators. Hence if `a` is `5`, `b` is `10`, `c` is `15` and `d` is `0` what are the truth values of the expressions?

```
c == a+b
a != 7
b <= a
a > 5
a+d >= c-b
d/a < c*b
```

44. Bracket the following logical expressions to show the order of evaluation of the operators. Hence if `a` is `5`, `b` is `10`, `c` is `15` and `d` is `0` what are the truth values of the expressions?

```
c == a+b || c == d
a != 7 && c >= 6 || a+c <= 20
!(b <= 12) && a % 2 == 0
!(a >5) || c < a+b
```

45. To what do the following expressions evaluate?

```
     17/3  17%3  1/2  1/2*(x+y)
```

46. Given the declarations:

```
float x;
int k, i = 5, j = 2;
```

To what would the variables `x` and `k` be set as a result of the assignments

```
k = i/j;
x = i/j;
k = i%j;
x = 5.0/j;
```

47. If `x` has the value `3.5` when the following statement is executed what value would be assigned to y?

```
if (x + 1 <= 3.6)
   y = 1.0;
else
   y = 2.0;
```

48. In words what do you think the effect of the following statement is intended to be? Why is the statement syntactically incorrect? How could it be changed to be syntactically correct? Has the change that you have made ensured that the statement actually carries out the logical effect you stated at the beginning?

```
if (x >= y)
   sum += x;
   System.out.println("x is bigger");
else
   sum += y;
   System.out.println("y is bigger");
```

49. Write an if-else statement which would add a variable `x` to a variable `possum` if `x` is positive and would add `x` to `negsum` if the variable `x` was negative.

50. Expand the solution to the previous question so that if `x` is positive then a variable `poscount` is incremented and if x is negative a variable `negcount` is incremented. If this was part of a program what would be sensible initializations to carry out?

51. It is decided to base the fine for speeding in a built up area as follows - 50 pounds if speed is between 31 and 40 mph, 75 pounds if the speed is between 41 and 50 mph and 100 pounds if he speed is above 50 mph. A programmer writing a program to automate the assessment of fines produces the following statement:

```
if (speed > 30)
   fine = 50;
else if (speed > 40)
   fine = 75;
else if (speed > 50)
```

```
   fine = 100;
```

Is this correct? What fine would it assign to a speed of 60 mph? If incorrect how should it be written?

52. Write a nested if-else statement that will assign a character grade to a percentage mark as follows - 70 or over: A, 60-69: B, 50-59: C, 40-49: D, 30-39: E, less than 30: F.

53. What is the major difference between a while statement and a do-while statement?

54. What would be output by the following segment of Java?

```
int i;
i = -12;
do
{
   System.out.println( i );
    i = i - 1;
}
while (i > 0)
```

55. What would be output by the following segment of Java?

```
int i;
for ( i = 1; i <= 12; i *= 2 )
   System.out.println( i );
```

56. What is printed by the following segment of Java?

```
int i;
for (i=1; i<20; i = i+3)
   System.out.println( i );
```

57. What would happen if the `i+3` in the update expression were changed to `i-3`?

58. An integer, x has a binary value (using 1 byte) of 10011100. What is the binary value of z after these statements:

```
int y = 1 << 7;
int z = x & y;
```

59. What will happen if you compile and execute an application with the following code in its main() method:

```
String s = new String( "Computer" );
if( s == "Computer" )
        System.out.println( "Equal A" );
if( s.equals( "Computer" ) )
        System.out.println( "Equal B" );
```

60. Consider this class example:

```
class MyPoint
{
   void myMethod()
   {
      int x, y;
      x = 5; y = 3;
      System.out.print( " ( " + x + ", " + y + " ) " );
      switchCoords( x, y );
      System.out.print( " ( " + x + ", " + y + " ) " );
   }
   void switchCoords( int x, int y )
   {
      int temp;
      temp = x;
      x = y;
      y = temp;
      System.out.print( " ( " + x + ", " + y + " ) " );
   }
}
```

What is printed to standard output if myMethod()is executed?

61. If `arr[]` contains only positive integer values, what does this function do?

```
public int guessWhat( int arr[] )
{
   int x= 0;
   for( int i = 0; i < arr.length; i++ ) x = x < arr[i] ? arr[i] : x;
   return x;
}
```

62. Consider the code below:

```
arr[0] = new int[4];
arr[1] = new int[3];
arr[2] = new int[2];
```

```
arr[3] = new int[1];
for( int n = 0; n < 4; n++ )
  System.out.println( /* what goes here? */ );
```

What expression should be written instead /* what goes here? */ to print the number of values in each row?

63. If `size = 4,` what is the row by row contents of `triArray:`

```
int[][] makeArray( int size)
{
   int[][] triArray = new int[size] [];
   int val=1;
   for( int i = 0; i < triArray.length; i++ )
   {
      triArray[i] = new int[i+1];
      for( int j=0; j < triArray[i].length; j++ )
        triArray[i][j] = val++;
   }
   return triArray;
}
```

64. Consider the code below:

```
public static void main( String args[] )
{
   int a = 5;
   System.out.println( cube( a ) );
}
int cube( int theNum )
{
   return theNum * theNum * theNum;
}
```

What will happen when you attempt to compile and run this code?

65. What would be printed by the code below if `val = 1?`

```
switch( val )
{
   case 1: System.out.print( "P" );
   case 2:
   case 3: System.out.print( "Q" );
      break;
   case 4: System.out.print( "R" );
   default: System.out.print( "S" );
}
```

66. What would be printed to standard output by the code fragment below?

```
outer: for( int i = 1; i <3; i++ )
{
   inner: for( j = 1; j < 3; j++ )
   {
      if( j==2 ) continue outer;
      System.out.println( "i = " +i ", j = " + j );
   }
}
```

67. In Java technology what expression can be used to represent number of elements in an array named `arr` ?

68. What is returned when the method `substring(2, 4)` is invoked on the string `"example"?` Include the answer in quotes as the result is of type `String.`

69. What gets printed when the following program is compiled and run?

```
public static void main(String args[])
{
      int i, j=1;
      i = (j > 1)? 2: 1;
      switch(i)
      {
        case 0: System.out.println(0); break;
        case 1: System.out.println(1);
        case 2: System.out.println(2); break;
        case 3: System.out.println(3); break;
      }
}
```

70. What happens upon invocation of a Java method?

71. What happens when the **new** Java operator is invoked?

72. What will happen when the following code snippet gets executed?
```
BigInteger big1 = new BigInteger("1");
BigInteger big2 = new BigInteger("2");
big1 = big2;
```

73. Is the following code correct and complete? Why?
```
public int sum2(int[] data)
{
  int sum = 0;
  for (int i = 0; i <= data.length; i++)
    sum += data[i];
}
```

74. How does a String differ from an array of characters?

75. What is a Java interface?

76. In what does an **interface** differ from a **class**?

77. Describe the two components of an interface definition.

78. Describe the restrictions which apply to Java interfaces.

79. When should interfaces be used in Java programs?

80. What is the purpose of Java packages?

81. What are local variables in Java?

82. What are instance variables in Java?

83. What are class variables in Java?

84. What does the following expression evaluate to?
```
-4 >>> 28
```

85. List the differences between methods and constructors.

86. Explain why the following sequence is correct or incorrect
```
short s = 20;
char c = s;
```

87. Explain why the following sequence is correct or incorrect
```
byte b = 20;
char c = b;
```

88. Explain why the following sequence is correct or incorrect
```
short s1 = 10;
short s2 = 20;
short result = s1*s2;
```

89. Given the following class declaration
```
public class MyClass
{
  public static void main(String arg)
  {
    MyClass mc = new MyClass( );
    System.out.println("First Argument is  : "+arg[0]);
  }
}
```
What happens when you try to compile and run `MyClass`?

90. What happens when you try to compile and the code below?
```
public class MyClass
{
  static int i = 10;
  public static void main(String[] arg)
  {
        static int i = 20;
        System.out.println("i is :"+i);
  }
```

```
}
```

91. How many `String` objects are created when we run the following code? Motivate your answer.

```
String s1,s2,s3,s4;
s1 = "Hello";
s2 = s1;
s3 = s2 + "Pal";
s4 = s3;
```

92. What is the output of the following code? Why?

```
int i = 10;
long l = 10L;
if( i == l )
   System.out.println("We are Equal");
```

93. What is the output of the following code? Why?

```
int i = 10;
char c = 10;
if( c == i)
   System.out.println("We are Equal");
```

94. What is the output of the following code? Why?

```
String s1 = "Null";
String s2 = "Null";
if( s1 == s2)
  System.out.println("We are Equal");
```

95. What is the output of the following code? Why?

```
String s1 = "Null";
String s2 = new String(s1);
if( s1 == s2)
  System.out.println("We are Equal");
```

96. What is the output of the following code? Why?

```
String s1 = "OK";
String s2 = new String(s1);
if( s1.equals(s2))
  System.out.println("We are Equal");
```

97. What is the output of the following code? Why?

```
Boolean b1 = new Boolean(true);
Boolean b2 = new Boolean(true);
if(b1.equals(b2))
  System.out.println("We are Equal");
```

98. To what values are local variables initialized?

Questions/Exercises for final

1. What are the differences between composition and inheritance?
2. What is polymorphism. Give a brief example.
3. What is the difference between overriding and overloading. Give brief examples.
4. Given classes A, B, and C, where B extends A, and C extends B, and where all classes implement the instance method void `doIt()` . How can the `doIt()` method in A be called from an instance method in C? Why?
5. What would be the result of compiling and running the following program?

```
// Filename: MyClass. java
public class MyClass {
  public static void main(String[] args) {
    C c = new C();
    System.out.println(c. max(13, 29));
  }
}
class A {
  int max(int x, int y) { if (x>y) return x; else return y; }
}
class B extends A{
  int max(int x, int y) { return super.max(y, x) - 10; }
}
class C extends B {
  int max(int x, int y) { return super.max(x+10, y+10); }
```

```
}
```

6. Which is the simplest expression that can be inserted at (1), so that the program prints the value of the text field from the `Message` class?

```
// Filename: MyClass.java
class Message {
   // The message that should be printed:
  String text = "Hello, world! ";
}
class MySuperclass {
  Message msg = new Message() ;
}
public class MyClass extends MySuperclass {
  public static void main(String[] args) {
    MyClass object = new MyClass();
    object.print();
  }
  public void print() {
    System.out.println( /* (1) WRITE THIS COMPLETED STATEMENT */ ) ;
  }
}
```

7. Which method declarations, when inserted at (7), will not result in a compile-time error?

```
class MySuperclass {
  public        Integer step1(int i)     { return 1; }     // (1)
  protected     String  step2(String str1, String str2) { return str1; }  // (2)
  public        String step2(String str1)    { return str1; }  // (3)
  public static String step2()        { return "Hi "; }  // (4)
  public MyClass      makeIt() { return new MyClass(); }           // (5)
  public MySuperclass makeIt2() { return new MyClass(); }          // (6)
}
public class MyClass extends MySuperclass {
  // (7) WRITE THIS METHOD DECLARATION
}
```

8. What would be the result of compiling and running the following program?

```
class Vehicle {
  static public String getModelName() { return "Volvo"; }
  public long getRegNo() { return 12345; }
}
class Car extends Vehicle {
  static public String getModelName() { return "Toyota"; }
  public long getRegNo() { return 54321; }
}
public class TakeARi de {
  public static void main(String args[]) {
    Car c = new Car();
    Vehi cle v = c;
    System.out.printl n("| " + v.getModelName() + "| " + c. getModel Name() +
  "| " + v.getRegNo() + "| " + c. getRegNo() + "| ");
  }
}
```

9. What would be the result of compiling and running the following program?

```
final class Item {
  Integer size;
  Item(Integer size) { this. size = size; }
  public boolean equals(Item item2) {
    if (this == item2) return true;
    return this.size.equals(item2.size);
  }
}
public class SkepticRide {
  public static void mai n(String[] args) {
    Item itemA = new Item(10);
    Item itemB = new Item(10);
    Object itemC = itemA;
    System.out.printl n("| " + itemA.equal s(itemB) +
      "| " + itemC.equal s(itemB) + "| ") ;
  }
}
```

10. Which constructors can be inserted at (1) in MySub without causing a compile-time error?

```
class MySuper {
  int number;
  MySuper(int i) { number = i ; }
```

```
}
class MySub extends MySuper {
  int count;
  MySub(int count, int num) {
    super(num);
    this.count = count;
  }
  // (1) WRITE CONSTRUCTOR NEEDED At THIS POINT
}
```

11. What will the following program print when run?

```
// Filename: MyClass.java
public class MyClass {
  public static void main(String[] args) {
    B b = new B("Test");
  }
}
class A {
  A() { this("1", "2"); }
  A(String s, String t) { this(s + t); }
  A(String s) { System.out.printl n(s); }
}
class B extends A {
  B(String s) { System.out.printl n(s); }
  B(String s, String t) { this(t + s + "3"); }
  B() { super("4"); };
}
```

12. Consider the following two class definitions.

```
class X {
  public double g(double x) {
    return f(x) * f(x);
  }
  public double f(double x) {
    return x + 1.0;
  }
}
class Y extends X {
  public double f(double x) {
    return x + 2.0;
  }
}
```

What will the following sequence of statements print?

```
Y y = new Y();
X x = y;
System.out.println(y.f(2.0));
System.out.println(x.f(2.0));
System.out.println(x.g(2.0));
```

13. Suppose a user executes the following `run()` method and types the numbers 5 and 22.

```
public static void run() {
    int a = IO.readInt();
    int b = IO.readInt();

    int m = 0;
    int n = b;
    while(m < n) {
        m += a;
        n -= a;
    }
    System.out.println(m - n);
}
```

Show all the values taken on by the variables of the program.

```
a
b
m
n
```

What does the method print?

14. Complete the below class method so that it reads two strings from the user and displays "yes"' if the second string includes only characters from the first, and "no"' if it contains any characters that the first does not include. For example, I should be able to execute your method and see the following. (Boldface indicates what the user types.)

brillig glib
yes

Or I might see the following. In this example, it prints "no" because broil includes the letter o, which does not occur in brillig.

brillig broil
no
```
public static void run() {

}
```
To accomplish this, you may find the following `String` instance methods useful.

`int length()`
: Returns the number of characters in the target string.

`char charAt(int i)`
: Returns the character at index `i`. For example, if `str` holds the string brillig, `str.charAt(2)` would return the character `'i'`.

`int indexOf(String s)`
: Returns the index where `s` occurs first within the string on which the method is called. If `s` occurs nowhere within the target string, the method returns -1. For example, if `str` holds the string brillig, `str.indexOf("il")` would return 2, since this is the index where il occurs first.

15. At right, write a definition of a new type, called `IntRange`, representing a range of integers. This class should support the following instance methods.

`IntRange(int start, int end)`
: (Constructor method) Sets up a new `IntRange` object representing the set of integers between `start` and `end`, including these two endpoints. The method may assume that the first parameter is below or equal to the second parameter.

`int getSize()`
: Returns the number of integers in the range.

`boolean contains(int i)`
: Returns `true` if `i` lies within the range.

The following example method, which uses the `IntRange` class you define, illustrates how the class should work.
```
public static void run() {
    IntRange range = new IntRange(2, 4);
    System.out.println(range.getSize());         // this prints ``3''
    System.out.println(range.contains(1));       // this prints ``false''
    System.out.println(range.contains(3));       // this prints ``true''
    System.out.println(range.contains(4));       // this prints ``true''
    System.out.println(range.contains(5));       // this prints ``false''
}
public class IntRange {



}
```

16. Suppose we have the following two class definitions.
```
class A {                class B extends A {
  public int f(int x) {    public int f(int x) {
    return 2 * x;            return 5 * x;
  }                        }
  public int g(int x) {    public int h(int x) {
    return f(x * 3);         return f(7 * x);
  }                        }
}                        }
```
And suppose we execute the following `run` method.
```
public static void run() {
  B b = new B();
  System.out.println(b.f(1) + " " + b.g(1) + " " + b.h(1));
  A a = b;
  System.out.println(a.f(1) + " " + a.g(1));
}
```
What would the method print?

17. Define a class `PassCount` to track whether all the students of a class have passed a test. It should support the following methods.

`PassCount()`
(Constructor method) Constructs an object representing an empty class.

`void addGrade(double grade)` Adds `grade` into the class.

`boolean isAnyFailing()` Returns `true` only if there is somebody in the class who received a grade of less than 60.

For example, if you defined this class properly, I should be able to write the following class to test it.
```
public class PassCountTest {
   public static void run() {
   PassCount a = new PassCount();
   addGrade(45.0);
   addGrade(76.0);
   System.out.println(a.isAnyFailing()); // should print "true"

   PassCount b = new PassCount();
   addGrade(60.0);
   System.out.println(b.isAnyFailing()); // should print "false"
   }
```

```
}
```

Note that, to accomplish this, a `PassCount` object need not remember every single grade --- that is, you do not need to use arrays: It just needs to remember whether any of the ones it has seen were below 60.

18. Define a class `NumberIterator` for iterating through a sequence of numbers. It should support the following methods.

`NumberIterator(int start, int stop)`
: (Constructor method) Constructs an object for iterating through the integers beginning at `start` and going up to `stop`. The constructor assumes that `start` is less than `stop`.

`boolean hasMoreNumbers()`
: Returns `true` if there are more numbers remaining in the sequence.

`int nextNumber()`
: Returns the current number in the sequence and steps the iterator forward, so that the next call to this method returns the following number in the sequence. This method initiates a `NoSuchElementException` if the sequence has no more elements remaining.

For example, if you defined this class properly, I should be able to write the following class to test it. When executed, its `run()` method would print "`5 6 7 8`".

```
public class NumberIteratorTest {
    public static void run() {
        NumberIterator it = new NumberIterator(5, 8);
        System.out.print(it.nextNumber());
        while(it.hasMoreNumbers()) {
            int i = it.nextNumber();
            System.out.print(" " + i);
        }
    }
}
```

19. Suppose we have the following two class definitions.

```
class P {
  public int f(int x) {
    return x + 1;
  }
  public int g(int x) {
    return f(x + 2);
  }
}
```

```
class Q extends P {
  public int f(int x) {
    return x + 4;
  }
  public int h(int x) {
    return f(x + 8);
  }
}
```

And suppose we execute the following `run` method.

```
public static void run() {
  P a = new P();
  Q b = new Q();
  P c = b;
  System.out.println(a.f(0) + " " + a.g(0));
  System.out.println(b.f(0) + " " + b.g(0) + " " + b.h(0));
  System.out.println(c.f(0) + " " + c.g(0));
}
```

What would the method print?

20. Suppose we have the class defined as below.

```
class C {
    static int y = 0;
    int z;

    C() {
        z = 0;
    }
    void incrX() {
        int x = 0;
        x++;
        IO.println(x);
    }
    void incrY() {
        y++;
        IO.println(y);
    }
    void incrZ() {
        z++;
        IO.println(z);
    }
}
```

```
public class Example {
    public static void run() {
        C a = new C();
        C b = new C();
        a.incrX();
        a.incrX();
        b.incrX();
        a.incrY();
        a.incrY();
        b.incrY();
        a.incrZ();
        a.incrZ();
        b.incrZ();
    }
}
```

What would the computer print when it executes the `Example` class's `run()` method?

21. A matrix is symmetric if, for each i and j, $a_{i,j} = a_{j,i}$. The following is an example of a symmetric matrix.

    0 23 45
    23 10 36
    46 36 20

    Another way of defining it: A symmetric matrix can be reflected across its main diagonal (top left corner to bottom right corner) to obtain the same matrix. Complete the following method so that it returns `true` only if the two-dimensional array parameter `mat` is symmetric. Your solution may assume that the matrix is square (as many columns as rows).

```
public static boolean isSymmetric(int[][] mat) {



}
```

    (Your solution shouldn't really be this long!)

22. Define a class `PigPen` for tracking the number of pigs in in a pen. It should support the following methods.

`PigPen(int pigs)`
: (Constructor method) Constructs an object representing a pig pen containing `pigs` pigs.

`boolean isEmpty()`
: Returns `true` if there are no pigs in the pen.

`void pigEnters()`
: Adds one to the number of pigs in the pen.

`void pigExits()`
: Subtracts one from the number of pigs in the pen.

For example, if you defined this class properly, I should be able to write the following class to test it.

```
public class PigPenTest {
    public static void run() {
        PigPen pen = new PigPen(2);
        pen.pigExits();
        System.out.println(pen.isEmpty()); // prints "false"
        pen.pigExits();
        System.out.println(pen.isEmpty()); // prints "true"
        pen.pigEnters();
        System.out.println(pen.isEmpty()); // prints "false"
    }
}
```

23. Write a class method named `mode` that takes an array of `int`s as a parameter and returns the integer that occurs in the array most frequently. For example, the following code fragment that uses your `mode` method should print 23.

```
int[] a = { 23, 34, 45, 23, 0, 23 };
System.out.println(mode(a));
```

    Your method should not call any other methods to accomplish this. It will need more than one loop to count the number of occurrences of each number in the array.

24. Consider the following Java program.

```
class A {
    int f() { return 1; }
}
class B extends A {
    int f() { return 0; }
}
class Main {
    public static void main(String[] args) {
        A a = new B();
        System.out.println(a.f());
    }
}
```

    What does this program print?

25. Consider the following program.

```
class Ident {
    int f(int x) { return x; }
    int g(int x) { return f(f(x)); }
}
class Square extends Ident {
    int f(int x) { return x * x; }
}
class Main {
    public static void main(String[] args) {
        Ident a = new Ident();
```

```
        Ident b = new Square();
        Square c = new Square();
        System.out.println(a.g(3) + " " + b.g(3) + " " + c.g(3));
    }
}
```

What does this program print?

26. Write an expression will extract the substring "kap", given the following declaration:
```
String str = "kakapo";
```

27. What will be the result of attempting to compile and run the following code?
```
class MyClass {
  public static void main(String[] args) {
    String str1 = "str1";
    String str2 = "str2";
    String str3 = "str3";
    str1.concat(str2) ;
    System.out.println(str3.concat(str1));
  }
}
```

28. What will be the result of attempting to compile and run the following program?
```
public class RefEq {
  public static void mai n(String[] args) {
    String s = "ab" + "12";
    String t = "ab" + 12;
    String u = new String("ab12");
    System.out.println((s==t) + " " + (s==u));
  }
}
```

29. What is a Java exception?

30. How can done deal with exceptional conditions in Java. Give a short example.

31. Describe `the thry-throw-catch` mechanism.

32. State the catch-or-declare rule.

33. How do exceptions propagate? Give a brief example.

34. What is the minimum a user-defined exception class must contain? Give a short example.

35. What is an *inner* class?

36. What is an *anonymous inner* class? Give a short example.

37. What is a local class? Give a short example.

38. What is a Java *component*? Give a short example.

39. What is a Java *container*? Give a short example.

40. What are the Basic steps in displaying Java Graphics?

41. What is a listener in Java?

42. What is the purpose of call-backs?

43. What is the role of the *model* in the MVC architecture?

44. What is the role of the *controller* in the MVC architecture?

45. What is the role of the *view* in the MVC architecture?

46. Briefly describe the Swing separable model architecture.

47. What is a Java `Enumeration`? Give a small example.

48. What is a *generic type* in Java? Give a small example.

49. Write a generic method which prints the values stored in a collection.

50. What is an *iterator* in Java? Give a small example.

51. How can one include primitive values in a Java `Collection`? Give a short example.

52. What is a Java Collection?
53. What are the main differences between classes `Vector` and `ArrayList`?
54. What is the purpose of software testing?
55. What is functional testing?
56. What defines a test case?
57. What must be considered when developing a test plan?
58. What is a unit test?
59. What is a test harness?
60. What is regression testing?
61. What is test coverage?
62. How can one get a program trace? (Hint: there are at least 2 ways)
63. What are the benefits of logging?
64. What are the shortcomings of logging?
65. What will the following program print when run?

```
public class UppTurn {
  public static void mai n(String[] args) {
    String str1 = "l ower", str2 = "LOWER", str3 = "UPPER";
    str1.toUpperCase();
    str1.replace("LOWER", "UPPER");
    System.out.printl n((str1. equals(str2)) + " " + (str1. equals(str3))) ;
  }
}
```

66. Describe the steps needed to read strings from a formatted sequential file.
67. What will the method `length()` in the class `File` return?
68. If `write(0x01234567)` is called on an instance of `OutputStream`, what will be written to the destination of the stream?
69. Given the following program:

```
import java.io.DataInputStream;
import java.io.EOFException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
public class Endings {
  public static void main(String[] args) {
    try {
      FileInputStream fos = new FileInputStream("info.dat") ;
      DataInputStream dis = new DataInputStream(fos);
      int i= dis.readByte();
      while (i != -1) {
        System.out.print((byte)i + "| ");
        i= dis.readByte();
      }
    } catch (FileNotFoundException fnf) {
      System.out.println("File not found");
    } catch (EOFException eofe) {
      System.out.println("End of stream");
    } catch (IOException ioe) {
      System.out.println("Input error");
    }
  }
}
```

What will happen when one attempts to compile and run it?

70. How many methods are defined in the `Serializable` interface?
71. Given the following code:

```
public class Person  {
  protected String name;
  Person() { }
```

```
  Person(String name) { this. name = name; }
}
```

```
import java.io. Serializable;
public class Student extends Person implements Serializable {
  private long studNum;
  Student(String name, long studNum) {
    super(name);
    this. studNum = studNum;
  }
  public String toString() { return "(" + name + ", " + studNum + ") "; }
}
```

```
import java.io.*;
public class RQ800_10 {
public static void main(String args[])
    throws IOException, ClassNotFoundException {
    FileOutputStream outputFile = new FileOutputStream("storage. dat");
    ObjectOutputStream outputStream = new ObjectOutputStream(outputFile);
    Student stud1 = new Student("Aesop", 100);
    System.out.print(stud1);
    outputStream. writeObject(stud1);
    outputStream. flush();
    outputStream. close();
    FileInputStream inputFile = new FileInputStream("storage. dat");
    ObjectInputStream inputStream = new ObjectInputStream(inputFile);
    Student stud2 = (Student) inputStream.readObject();
    System.out.println(stud2);
    inputStream. close();
  }
}
```

What will happen when one attempts to compile and run it?

72. What will happen when one attempts to compile and run the following code:

```
import java.util. ArrayList;
import java.util. Collections;
import java.util. List;
public class WhatIsThis {
  public static void main(String[] args) {
    List<StringBuilder> list = new ArrayList<StringBuilder>();
    list.add("B");
    list.add("A");
    list.add("C");
    Collections.sort(list, Collections.reverseOrder());
    System.out.println(list.subList(1, 2));
  }
}
```

73. How can one achieve object persistence in Java?
74. What is a Java `Buffer`?
75. What are the benefits of `Buffer` *views*?
76. What is the purpose of a direct `Buffer`? Give a brief example.
77. What is a Java `Channel`?
78. When does the execution of a thread end?
79. How can the priority of a thread be set?
80. Describe the life cycle of a thread.
81. How should one correctly terminate a Java thread?
82. What is a Java Collection?
83. What are the main restrictions which apply to applets?
84. What are the differences between a standalone application and an applet?
85. Describe the life cycle of an applet.
86. What should be written in each of an applet's predefined methods?

87. What are the top level containers for GUIs in Java?
88. What are Java threads?
89. What are the differences between threads and operating system tasks?
90. What is a factory method? Give a brief example.

Last revisited on 11/19/2013 11:28:26