Exercises – Java Introduction

- Review Questions
- Coding Questions

Review Questions:

- **List the following from Robot Manager (below)**
  - 1. For all classes, give
    - the class name
    - the access modifier
  - 2. For all methods give
    - the method name
    - access modifier
  - 3. For each method
    - state the line number where the method body begins and ends.
    - list the method parameters if any
  - 4. Method Calls
    - on what line numbers are methods from the Robot class called?
  - 5. Objects
    - Are any objects created in the program?
    - If so, where? Give the line numbers.
  - 6. Variables:
    - List all variables types and names.
    - Identify any object reference vairables
    - What is the difference between a local and instance variable?
  - 7. Statements:
    - List any expression statemets, declaration statements or control flow statements
  - 8. The JVM
    - What is the difference between the stack and the heap??
    - What is Garbage Collection?
  - 9. OOP
    - What does Inheritance allow for?
    - What is a subclass?
    - What is a superclass?
  - 10. OOP
    - What does Polymorphism allow for ?
    - Explain how different implementations of the same method can be invoked from a superclass?

- 11. Access Modifiers:
  - List these in order of accessibility: private, public, default, protected?

- 12. Errors:
  - What is the difference between a compiler and run-time error?

CODE:

```
1. public class RobotManager {
2. public static void main(String[] args) {
3.    Robot robot;
4.    int nargs=args.length;
5.    if(nargs <= 1){
6.       System.out.println("There must be at least two arguments in the
          command!");
7.            System.out.println("Example: java RobotManager Mary Kaan");
8.            System.exit(0);
9.    } else {
10.      for(int i=1; i<nargs; i++){
11.           robot = new Robot();
12.           robot.setName(args[i]);
13.           robot.sayHelloTo(args[0]);
14.      }
15.    }
16. }
17.     }
18.
18. class Robot {
19. private String myName = "nobody";
20. public void setName(String name) {
21.    myName = name;
22. }
23. public void sayHelloTo(String name) {
24.   System.out.println("Hello " + name + "!");
25.   System.out.println("I'm Robo" + myName + ".");
26. }
27. }
```

Coding Questions:

1. Run Robot Manager from the command line:

- using two arguments:

  `java RobotManager Ginny 420`

- Now try it with three arguments

  `java RobotManager Ginny 420 421`

- And then

  `java RobotManager Ginny 420 number1`

- What happens?
- Try running it with only one command line argument

  `java RobotManager Ginny`

- What happens? Hint: Look at the if statement (line 5)

2. Modifying a Method:

- Modify the text on line 24

  From: **`"Hello "`** to **`"Hello, how are you "`**

- Re compile and run the program
- Modify the text on lines 24 and 25: Switch ***`name`*** and ***myname***
- Re compile and run the program

3. Creating a new method:

- Add the following method to Robot Class:

  Method name:**`sayHelloAgain`**
  Return Type: **Void**
  Arguments: none
  Body: write the code to print out **`"hello again"`** in the method body

- Compile – make sure there are no errors

4. Call the new method from the main method:

- Move the curser to the end of line 13 and press return to create a new line below
- On the new line below: call the new method you have just created.
    - Tip: you will use the **"robot"** object
    - Tip: to call a method you use the following syntax:
      object.methodname();
- Re compile and run the program
- By analysing the command line output - Did you see that your method was called?

Solutions to Java Introduction Exercises

Exercises 1:

List the following from Robot Manager:

- For all classes, give the
  - class name: There are two classes in the Robot Manager code example, namely the *RobotManager* class and the *Robot* class.
  - the access modifiers: The *RobotManager* class is public, while the *Robot* class has a default access modifier since no access modifier is specified.
- For all methods, give the
  - method name: There is a single method in the *RobotManager* class called *main.* There are two methods is the *Robot* class, set*Name* and *sayHelloTo.*
  - access modifier: All three methods are declared as public methods.
- For each method state the line number where the method body begins and ends
  - *main* method: lines 2 and 16
  - *setName* method: lines 20 and 22
  - *sayHelloTo* method: lines 23 and 26
- For each method from the Robot class called state the line number:
  - line 11: You probably missed this one. On this line the code “new Robot()” calls the Robot class constructor method which creates a new object instance of the class. Don't worry if you missed this one as we will be converting constructors later!!!
  - line 12 : Robot.setName() is called
  - line 13: Robot.sayHello() is called
- Objects:
  - Are any objects created in the program? If so give the line numbers.
    - Yes the number of objects created depends on the input value given on the command line when running the program. On line 11 the **new** keyword signifies the creation of a new object of type Robot (*new Robot()*). The object is created inside a for loop. The code will be repeated if more than two strings are entered on the command line. For example, if there are 3 strings entered two objects will be created, if there are 4 strings three objects will be created etc.
- Variables:
  - List all variables types and names (the line indicates where they are being declared):
    - line 2: *args* of type String array (part of the method argument list)
    - line 3: *robot* of type Robot
    - line 4: *nargs* of type int
    - line 10: *i* of type int
    - line 19: *myName* of type String

        - line 20 + 23: *name* of type String (part of the method argument list)
    - Identify all object references
        - robot is an object reference variable, so are all the variables from above that are of type String
    - What is the difference between local and instance variables
        - A variable declared inside a method is called a local variable (e.g. robot or nargs)
        - A variable declared outside of a method is called a class variable or an instance variable. (e.g. myName)
- Statements
    - List any expression statements
        - lines 4, 6, 7,8,10 (within control flow statement),11,12,13,19, 21, 24, 25
    - declaration statements
        - lines 3, 4, 10 (within control flow statement), 19,
    - control flow statements
        - lines 5, 9, 10
- The JVM
    - Heap vs Stack
        - The Heap is where memory is allocated for objects of the Java program, All the objects live on the heap
        - The Java Stacks contain the state of Java method invocations The state of a method invocation is called its stack frame
    - GC
        - Memory management in Java applications is handled by the garbage collector (unlike C/C++). Garbage collection is the process of automatically freeing objects that are no longer referenced by the program. Garbage collection relieves a developer from the burden of freeing allocated memory. When an object is no longer referenced (i.e. needed) by the program, the heap space it occupies can be recycled (by the garbage collector) so that the space is made available for subsequent new objects.
    - OOP
        - Inheritance
            - Allows for the derivation of a class (subclass) from another class (superclass).
            - The class from which you derive your class is called the superclass
            - A subclass inherits the state (variables) and the behaviour (methods) of the class from which it is derived.
        - Polymorphism

- allows an object of a superclass to refer to an object of any subclass
- see example on slide 40

- Access Modifiers
  - public, protected, default, private
- Errors
  - Compiler errors are caught at compilation time by the compiler
  - RuntimeErrors are caught when the program runs

Exercises – Data Types and Operators

- Review Questions 1 (Data Types and Operators)
- Review Questons 2 (More on Data Types)
- Review Questions 3 (More on Operators)
- Coding Questions

Review Questions 1 (Data Types and Operators)

1. Which of the following are invalid variable names in Java? (Choose all that apply.)

A. $char

B. 1MyNumber

C. case

D. _int

2. Consider the following line of code:

```
short    ohMy;
```

What is the range of values that could be assigned to the variable ohMy?

A. 0 to 2^16 – 1

B. 0 to 2^15 – 1

C. –2^15 – 1 to 2^15 – 1

D. –2^16 – 1 to 2^16 – 1

E. –2^15 to 2^15 – 1

F. –2^15 to 2^15

3. Consider the following line of code:

```
char    ohMy;
```

What is the range of values that could be assigned to the variable ohMy?

A. 0 to 2^16 – 1

B. 0 to 2^15 – 1

C. –2^15 – 1 to 2^15 – 1

D. –2^16 – 1 to 2^16 – 1

E. –2^15 to 2^15 – 1

F. –2^15 to 2^15

4. Consider the following line of code:

```
byte    ohMy;
```

What is the range of values that could be assigned to the variable ohMy?

A. 0 to 2^16 – 1

B. 0 to 2^8 – 1

C. –2^7 to 2^7 – 1

D. –2^7 to 2^7

E. –2^15 to 2^15 – 1

F. –2^8 to 2^8 – 1

5. Which of the following statements would not produce a compile error?

A. char my_char = 'c';

B. char your_char = 'int';

C. char what = 'Hello';

D. char what_char = "L";

E. char ok = '\u3456';

6. Consider the following declaration:

boolean iKnow;

The variable iKnow will be automatically initialized to which of the following?

A. true

B. false

7. Consider the following piece of code:

float luckyNumber = 1.25;

System.out.println ("The value of luckyNumber: " + luckyNumber);

What is the result?

A. The value of luckyNumber::.

B. The value of luckyNumber: 1.25.

C. This piece of code would not compile.

D. This piece of code would compile, but give an error at execution time.

8. Consider the following code fragment:

```
public class Unary{
  public static void main(String[] args) {
    int x = 7;
    int y = 6*x++;
    System.out.println (" y= " + y);
     int a = 7;
     int b = 6*++a;
      System.out.println (" b= " + b);
  }
}
```

What is the output of this code fragment?

A. y= 42

b= 48

B. y= 48

b= 48

C. y= 48

b= 42

D. y= 42

b= 42

9. Consider the following code fragment:

```
int  x;

int  a = 5;

int  b= 8;

x =  ++a + b++;
```

After the execution of this code fragment, what is the value of x?

A. 13

B. 14

C. 15

D. Compilation fails.

10. Which of the following expressions are illegal? (Choose all that apply.)

A. int x = 9;

B. int y = !x;

C. double z = 9.00 >> 2;

D. int i = ^7;

11. Consider the following code fragment:

```
1.  public class Unary{
2.     public static void main(String[] args) {
3.       byte x = 7;
4.       byte y = 6*x++;
5.       byte z = x*y;
6.       System.out.println ("z: " + z);
7.     }
8.  }
```

What is the output of this code fragment?

A. z: 42

B. z: 48

C. The code will not compile due to line 4.

D. The code will compile, but will generate a runtime error.

12. Consider the following code fragment:

```
int x = 9;

int y = -2;

System.out.println("output: " + x%y);
```

What is the output of this code fragment?

A. -1;

B. 1;

C. 4.5;

D. 4

13. Consider the following code fragment:

```
1. public class Question{
2.     public static void main(String[] args) {
3.         byte x = 21;
4.         byte y = 13;
5.         int z = x^y;
6.         System.out.println(z);
```

```
7.    }
```

```
1.  }
```

```
2.
```

What is the result of this code fragment?

A. 24

B. 29

C. 21

D. 13

E. A compiler error occurs at line 5.

14. Consider the following code fragment:

```
1. public class LogicTest{
2.    public static void main(String[] args) {
3.       int i = 5;
4.       int j = 10;
5.       int k = 15;
6.       if ( (i < j) || ( k-- > j) ) {
7.             System.out.println("First if, value of k: " + k);
8.       }
9.       if ( (i < j) && ( --k < j) ) {
```

```
10.             System.out.println("Second if, value of k: " + k);
11.         }
12.         System.out.println("Out of if, k:" + k);
13.       }
14.    }
```

What is the output of this code fragment?

A. First if, value of k: 14

Out of if, k: 13

B. First if, value of k: 15

Out of if, k: 14

C. First if, value of k: 15

Out of if, k: 13

15. Consider the following code fragment:

```
1. public class LogicTest{
2.     public static void main(String[] args) {
3.         int i = 5;
4.         int j = 10;
```

```
5.      int k = 15;
6.      if ( (i < j) || ( k-- > j) ) {
7.          System.out.println("First if, value of k: " + k);
8.      }
9.      if ( (i > j) && ( --k < j) ) {
10.         System.out.println("Second if, value of k: " + k);
11.     }
12.     System.out.println("Out of if, k:" + k);
13.  }
14. }
```

What is the output of this code fragment?

A. First if, value of k: 14

Out of if, k: 13

B. First if, value of k: 15

Out of if, k: 14

C. First if, value of k: 15

Out of if, k: 13

D. First if, value of k: 15

Out of if, k: 15

16. Consider this code. What is the result?

```
1. class CodeWalkOne {
2.     public static void main(String [] args) {
3.       int [] counts = {1,2,3,4,5};
4.       counts[1] = (counts[2] == 2) ? counts[3] : 99;
5.       System.out.println(counts[1]);
6.     }
7. }
```

A. Compilation fails on line 4.

B. 3

C. 4

D. 2

E. 99

Review Questions 2 (More on Data Types)

1. A signed data type has an equal number of non-zero positive and negative values available.

A. True

B. False

2. Choose the valid identifiers from those listed here. (Choose all that apply.)

A. BigOlLongStringWithMeaninglessName

B. $int

C. bytes

D. $1

E. finalist

3. Consider the following line of code:

```
int[] x = new int[25];
```

After execution, which statements are true? (Choose all that apply.)

A. x[24] is 0

B. x[24] is undefined

C. x[25] is 0

D. x[0] is null

E. x.length is 25

4. What is the range of values that can be assigned to a variable of type short?

A. Depends on the underlying hardware

B. 0 through 2^16 − 1

C. 0 through 2^32 − 1

D. −2^15 through 2^15 − 1

E. −2^31 through 2^31 − 1

5. What is the range of values that can be assigned to a variable of type byte?

A. Depends on the underlying hardware

B. 0 through 2^8 − 1

C. 0 through 2^16 − 1

D. −2^7 through 2^7 − 1

E. −2^15 through 2^15 − 1

6. What happens when you try to compile and run the following code?

```
public class Q15 {

  static String s;

  public static void main(String[] args) {

    System.out.println(“>>” + s + “<<”);

  }
```

}

A. The code does not compile

B. The code compiles, and prints out >><<

C. The code compiles, and prints out >>null<<

7. Which of the following are legal? (Choose all that apply.)

A. int a = abcd;

B. int b = ABCD;

C. int c = 0xabcd;

D. int d = 0XABCD;

E. int e = 0abcd;

F. int f = 0ABCD;

8. Which of the following are legal? (Choose all that apply.)

A. double d = 1.2d;

B. double d = 1.2D;

C. double d = 1.2d5;

D. double d = 1.2D5;

9. Which of the following are legal?

A. char c = 0x1234;

B. char c = \u1234;

C. char c = '\u1234';

Review Questions 3 (More on Operators)

1. After execution of the following code fragment, what are the values of the variables x, a, and b?

```
1. int x, a = 6, b = 7;

2. x = a++ + b++;
```

A. x = 15, a = 7, b = 8

B. x = 15, a = 6, b = 7

C. x = 13, a = 7, b = 8

D. x = 13, a = 6, b = 7

2. Which of the following expressions are legal? (Choose all that apply.)

A. int x = 6; x = !x;

B. int x = 6; if (!(x > 3)) {}

C. int x = 6; x = ~x;

3. Which of the following expressions are legal? (Choose all that apply.)

A. String x = "Hello"; int y = 9; x += y;

B. String x = "Hello"; int y = 9; if (x == y) {}

C. String x = "Hello"; int y = 9; x = x + y;

D. String x = "Hello"; int y = 9; y = y + x;

4. What is -8 % 5?

A. -3

B. 3

C. -2

D. 2

5. What is 7 % -4?

A. -3

B. 3

C. -4

D. 4

6. What results from running the following code?

```
1. public class Xor {
2.   public static void main(String args[]) {
3.     byte b = 10; // 00001010 binary
4.     byte c = 15; // 00001111 binary
5.     b = (byte)(b ^ c);
6.     System.out.println("b contains " + b);
7.   }
8. }
```

A. The output: b contains 10

B. The output: b contains 5

C. The output: b contains 250

D. The output: b contains 245

7. What results from attempting to compile and run the following code?

```
1. public class Conditional {
2.   public static void main(String args[]) {
3.     int x = 4;
4.     System.out.println("value is " +
5.         ((x > 4) ? 99.99 : 9));
6.   }
7. }
```

A. The output: value is 99.99

B. The output: value is 9

C. The output: value is 9.0

D. A compiler error at line 5

8. What does the following code do?

```
Integer i = null;

if (i != null & i.intValue() == 5)

  System.out.println("Value is 5");
```

A. Prints "Value is 5".

B. Throws an exception.

9. Suppose ob1 and ob2 are references to instances of java.lang.Object. If (ob1 == ob2) is false, can ob1.equals(ob2) ever be true?

A. Yes

B. No

10. When a byte is added to a char, what is the type of the result?

A. byte

B. char

C. int

D. short

E. You can’t add a byte to a char.

11. When a short is added to a float, what is the type of the result?

A. short

B. int

C. float

1. You can’t add a short to a float.
2.

12. Which statement is true about the following method?

```
int selfXor(int i) {
  return i ^ i;
}
```

A. It always returns 0.

B. It always returns 1.

C. It always an int where every bit is 1.

D. The returned value varies depending on the argument.

13. Which of the following operations might throw an ArithmeticException?

A. +

B. -

C. *

D. /

E. None of these

14. What is the return type of the instanceof operator?

A. A reference

B. A class

C. An int

D. A boolean

15. Which of the following may appear on the left-hand side of an instanceof operator?

A. A reference

B. A class

C. An interface

D. A variable of primitive type

16. Which of the following may appear on the right-hand side of an instanceof operator? (Choose all that apply.)

A. A reference

B. A class

C. An interface

D. A variable of primitive type

E. The name of a primitive type

Coding Questions:

Code Question1:

```
1. public class InitialTest {
2.      int x;
3.   public static void main(String[] args) {
4.       new InitialTest().printIt();
5.    }
6.    public void printIt(){
7.       int y;
8.       int z;
9.       y=2;
10.      System.out.println(x +" "+ y);
11.   // System.out.println(z);
12.   }
13. }
```

Run and compile

Uncomment line 11 and try to compile

The code will generate a compiler error because you are using the

local variable z without initializing it. Note that lines 7 and 8 did not generate compiler errors. That means you do not have to initialize a local variable in the same statement where you declare it.

However, you must initialize it before using it.

Code Question 2:

Run and Compile the CodeWalkOne class.

```
1.class CodeWalkOne {
2.      public static void main(String [] args) {
3.       int [] counts = {1,2,3,4,5};
4.       counts[1] = (counts[2] == 2) ? counts[3] : 99;
5.       System.out.println(counts[1]);
6.    }
7. }
```

Change the value of int [] counts = {1,2,3,4,5}; to int [] counts = {1,2,2,4,5};
Compile and run. What is the result ? Why?

Write code to
Declare a second int array
called oneTwoThree of size 3.
Fill it with numbers 1 ,2 and 3.
Print out the first value in the array. Print out the last value
Run and compile

Code Question 3:

Write a simple program that has a single class and main method.

That prints out the values of the following variables:

```
byte a = 70;

byte b = 5;

byte c = (byte) (a*b);
```

Do you get strange results??

The accurate value of c should be 350, which in binary format is 101011110. However, since c is

of type byte, only the lower 8 bits, 01011110, would be counted. Therefore, the value actually stored

in the variable c would be 94 instead of 350.

Now try printing out the vlaues of these variables:

int x = 70000;

int y = 70000;

int z = x*y;

Can you figure out the result in z?

Solutions to Exercsies – Data Types and Operators

Review Questions 1 (Data Types and Operators)

1. Answer: B and C

   B is invalid because it starts with a number, and C is invalid because it is a keyword and thuscan-not be a variable name. The first character of an identifier must be a letter, a dollar sign ($), or an underscore (_).

2. Answer: E

   The data type short is signed and 16 bits in size.

3. Answer: A

   The data type char is unsigned and 16 bits in size.

4. Answer: C

   The data type byte is unsigned and 16 bits in size.

5. Answer: A and E

   B, C, and D will have compiler errors because a char literal is represented by a single character enclosed in single quotes unless it's being represented as a Unicode value.

6. Answer: B

   A boolean type is initialized to false by default.

7. Answer: C

   The value 1.25 is a double, and therefore cannot be assigned to a float variable.

8. Answer: A

   If the increment (or decrement) operator is followed by the operand (e.g. ++x), then it operates

before the operand takes part in the rest of the expression. On the other hand, if the operand is followed by the operator (e.g. x++), then the operand takes part in the expression before the operator operates on it.

9. Answer: B

If the increment (or decrement) operator is followed by the operand (e.g. ++x), then it operates before the operand takes part in the rest of the expression. On the other hand, if the operand is followed by the operator (e.g. x++), then the operand takes part in the expression before the operator operates on it.

10. Answer: B, C, and D

The ! operator cannot be applied to an int, and the >> operator cannot be applied to a double. The operator ^ is not a unitary operator.

11. Answer: C

By arithmetic promotion, the right side of line 4 would be an integer. An integer cannot be assigned to a byte without explicit conversion.

12. Answer: B

The result of a % operation gets the sign of the numerator.

13. Answer: A

```
  21 = 00010101
^ 13 = 00001101
  --------------------
  00011000 = 24
```

The OR operation would produce 00011101 = 29.

14. Answer: B

In an OR (||) test, if the first condition is true, the second condition is not tested. In an AND (&&) test, if the first condition is true, the second condition will also be tested.

15. Answer: D

In an OR (||) test, if the first condition is true, the second condition is not tested. In an AND (&&) test, if the first condition is false, the second condition will not be tested.

16. Answer: E

E is the correct answer because the ternary test in line 4 fails, and as a result counts[1] is set equal to 99, which is printed in line 5.

Review Questions 2 (More on Data Types)

1.

B. The range of negative numbers is greater by one than the range of positive numbers.

2.

A, B, C, D, E. All of the identifiers are valid. An identifier begins with a letter, a dollar sign, or an underscore; subsequent characters may be letters, dollar signs, underscores, or digits. And of course keywords and their kin may not be identifiers.

3.

A, E. The array has 25 elements, indexed from 0 through 24. All elements are initialized to 0.

4.

D. The range for a 16-bit short is −215 through 215 − 1. This range is part of the Java specification, regardless of the underlying hardware.

5.

D. The range for an 8-bit byte is −27 through 27 −1. Table 1.3 lists the ranges for Java's integral primitive data types.

6.

C. The code compiles without error. At static initialization time, s is initialized to null (and not to a reference to an empty string, as suggested by C).

7.

C, D. The characters a–f and A–F may be combined with the digits 0–9 to create a hexadecimal literal, which must begin with 0x.

8.

A, B. The d suffix in option A and the D suffix in option B are optional. Options C and D are illegal because the notation requires e or E, not d or D.

9.

C. A Unicode literal character must be enclosed in single quotes and must begin with \u.

Review Questions 3 (More on Operators)

1.

C. The assignment statement is evaluated as if it were

```
x = a + b; a = a + 1; b = b + 1;
```

So the assignment to x is made using the sum of 6 + 7, giving 13. After the addition, the values of a and b are incremented; the new values, 7 and 8, are stored in the variables.

2.

B, C. In option A, the use of ! is illegal because x is of int type, not boolean. In option B, the comparison operation is valid, because the expression (x > 3) is a boolean type and the ! operator can properly be applied to it. In option C, the bitwise inversion operator is legally applied to an integral type.

3.

A, C. Options A and C are equivalent. They both add a String to an int; the resulting String is assigned to x, whose type is String, so the code is legal. Option B doesn't compile because an int and a String may not be compared. C is illegal because the last statement tries to assign a String to an int.

4.

A. When doing modulo arithmetic with a negative left-hand operand, the result is the negative of what it would be if both operands were positive. So -8 % 5 is the negative of 8 % 5.

5.

B. When doing modulo arithmetic with a negative right-hand operand, the result is the same as it would be if the right-hand operand were positive. So 7 % -4 is the same as 7 % 4.

6.

B. The XOR operator produces a 1 bit in any position where the operand bits are different. So

00001010 ^ 00001111 is 00000101, which is 5.

7.

C. In this code, the optional result values for the conditional operator, 99.99 (a double) and 9 (an int), are of different types. The result type of a conditional operator must be fully determined at compile time, and in this case the type chosen, using the rules of promotion for binary operands, is double. Because the result is a double, the output value is printed in a floating-point format.

If the two possible argument types had been entirely incompatible—for example, (x > 4) ? "Hello" : 9—then the compiler would have issued an error at that line.

8.

B. The & operator does not short circuit. Even though the left-hand operand is null, the right hand operand is still evaluated. Attempting to call the intValue() method on null results in a NullPointerException.

9.

B. The Object class' equals() method just does an == check, so if (ob1 == ob2) is false, then

ob1.equals(ob2) will always be false.

10.

C. Byte, short, and char operands are widened to ints before the addition is performed. The result type is the same as the operands: int.

11.

C. When a short is added to a float, the narrower data type (short) is widened to match the wider type (float), and the result is a float.

12.

A. Any value XOR itself is always 0.

13.

D. Only non-floating division can throw an ArithmeticException.

14.

D. The instanceof operator generates a boolean value.

15.

A. Only references may be the left-hand operands of instanceof.

16.

B, C. Classes, interfaces, arrays of classes, and arrays of interfaces may be the right-hand operands of instanceof.

Exercises and Solutions – Classes, Methods and Interfaces

- Review Questions (Methods, Classes and Interfaces)
- Coding Questions
  - Methods
  - Static Code
  - Vaarg Methods
  - Nested Classes
  - Classes defined inside Methods
  - Enums
  - Constructors and Inheritance

Review Questions 1:

1. Consider the following code:

```
1. class MyClass {
2. String hello = "Hello, Dear.";
3.     void printMessage() {
4.       System.out.println(hello);
5.     }
6. }
7. class TestMyClass {
8.       public static void main(String[] args) {
9.       MyClass mc = new MyClass();
10.        mc.printMessage();
11.     }
12.     }
```

What is the output of this code? (Choose one.)

A. A compiler error occurs at line 9 because no such method (constructor) for class MyClass has been declared.

B. A runtime exception occurs at line 4.

C. The code compiles and executes fine, and the output is Hello, Dear.

D. The code compiles and executes fine, and there is no output.

2. Consider the following code:

```
public class MyOuterClass {
    public static class MyNestedClass {
    }
}
```

Which of the following is a correct statement to instantiate MyNestedClass from a class outside of MyOuterClass? (Choose all that apply.)

A. MyNestedClass mn = new MyOuterClass.MyNestedClass();

B. MyOuterClass.MyNestedClass mn = new MyOuterClass.MyNestedClass();

C. MyOuterClass.MyNestedClass mn = new MyNestedClass();

D. MyOuterClass mo = new MyOuterClass();
MyOuterClass.MyNestedClass mn = mo.new MyNestedClass();

3. Consider the following code:

```
1. class A {
2.       A(String message) {
3.       System.out.println(message + " from A.");
4.    }
5. }
6.    class B extends A{
7.       B() {
8.            System.out.println("Hello from B.");
9.       }
```

```
10. }
11. class RunSubClass {
12.     public static void main(String[] args) {
13.         B b = new B();
14.     }
15. }
```

What is the output of this code?

A. Hello from B.

B. A compiler error occurs at line 2.

C. A compiler error is triggered by the call made at line 13.

D. It compiles fine but throws a runtime exception.

4. Which of the following is true about an interface? (Choose all that apply.)

A. You can declare a method with the private modifier in an interface.

B. You must use the public modifier while declaring a method in an interface.

C. You can declare variables inside an interface, and change their values in a class that implements the interface.

D. You can declare variables inside an interface, but you cannot change their values in a class that implements the interface.

E. You cannot declare a variable inside an interface.

5. Consider the following code:

```
1. class MyClass {
2. int i = 5;
3. static int j = 7;
```

```
4. public static void printSomething () {
5.      System.out.println("i: " + i);
6.      System.out.println("j: " + j);
7. }
8. }
```

What is the result if the printSomething() method of MyClass is called from another class?

A. i: 5
   j: 7

B. A compiler error occurs at line 5.

C. A compiler error occurs at line 6.

D. A runtime exception is thrown.

6. Consider the following class definition:

```
public class SubClass extends SuperClass {
public      SubClass ( int i){
}
public      SubClass(int i, int j) {
    super(i, j);
}
public SubClass (int i, int j, int k) {
}}
```

Which of the following forms of constructors must exist in the SuperClass? (Choose all that apply.)

A. SuperClass(int i) { }

B. SuperClass(int i, int j) { }

C. SuperClass() { }

D. SuperClass(int i, int j, int k) { }

7. Consider the following class definition:

```
1. class MyClass {
2. MyClass (int i ) {
3. }
4. void printTheThing(String message) {
5.       System.out.println(message);
6. }
7. }
8. class Test {
9. public static void main(String[] args) {
10.      MyClass myClass = new MyClass ( );
11.      myClass.printTheThing("Hello, I did not crash!");
12. }
13. }
```

Which of the following is a true statement about this code?

A. A compiler error occurs at line 10.

B. A compiler error occurs at line 2.

C. The code compiles but generates a runtime exception.

D. The code compiles and runs fine and produces the output Hello, I did not crash!.

8. Consider the following code fragment:

```
class SuperClass {
   SuperClass(){}
   SuperClass(int i) {
    System.out.println ("The value of i is " + i);
  }
}
 class SubClass extends SuperClass {
    SubClass(int j) {
     System.out.println ("The value of j is " + j );
      super(j);
    }
}

class Test {
  public static void main(String args[]) {
    SubClass sub = new SubClass(5);
  }
}
```

What output is generated when the class Test is run?

A. The value of i is 5.

B. The value of j is 5.

C. The value of i is 5 The value of J is 5.

D. A compiler error occurs.

E. An exception is thrown at execution time.

9. Which of the following are true statements? (Choose all that apply.)

A. A class can inherit from more than one class by using the keyword extends.

B. An interface can inherit from more than one interface by using the keyword extends.

C. A class can inherit from more than one interface by using the keyword extends.

D. A class can inherit from more than one interface by using the keyword implements.

10. Which of the following are illegal enum definitions?

A. enum Day {Sunday, Monday, Tuesday}

B. enum Day {Sunday, Monday, Tuesday,

```
    private String holiday;
}
```

C. enum Day {Sunday, Monday, Tuesday;

```
    private String holiday;
}
```

D. enum Day {private String holiday;

```
    Sunday, Monday, Tuesday;
}
```

E. enum Day {Sunday, Monday, Tuesday;

```
    private String holiday;
    Day(){
      System.out.println("Hello");
    }
```

}

11. Consider Listing 3-13. What is the output?

A. 5 11

B. 11 11

C. 5 5

D. Compilation fails at line 17.

E. Compilation fails at lines 5 and 6.

12. Consider the following class definition:

```
1. public class Test extends Base {
2.    public Test(int j) {
3.    }
4.    public Test(int j, int k) {
5.      super(j, k);
6.    }
7. }
```

Which of the following are legitimate calls to construct instances of the Test class? (Choose all that apply.)

A. Test t = new Test();

B. Test t = new Test(1);

C. Test t = new Test(1, 2);

D. Test t = new Test(1, 2, 3);

E. Test t = (new Base()).new Test(1);

13. Consider the following class definition:

```
1. public class Test extends Base {
2.    public Test(int j) {
3.    }
4.    public Test(int j, int k) {
5.      super(j, k);
6.    }
7. }
```

Which of the following forms of constructor must exist explicitly in the definition of the Base class? Assume Test and Base are in the same package. (Choose all that apply.)

A. Base() { }

B. Base(int j) { }

C. Base(int j, int k) { }

D. Base(int j, int k, int l) { }

14. Consider the following definition:

```
1. public class Outer {
2.    public int a = 1;
3.    private int b = 2;
4.    public void method(final int c) {
5.      int d = 3;
6.      class Inner {
```

```
7.      private void iMethod(int e) {
8.
9.      }
10.   }
11.  }
12. }
```

Which variables can be referenced at line 8? (Choose all that apply.)

A. a

B. b

C. c

D. d

E. e

15. Which of the following statements are true? (Choose all that apply.)

A. Given that Inner is a nonstatic class declared inside a public class Outer and that appropriate constructor forms are defined, an instance of Inner can be constructed like this: new Outer().new Inner()

B. If an anonymous inner class inside the class Outer is defined to implement the interface ActionListener, it can be constructed like this: new Outer().new ActionListener()

C. Given that Inner is a nonstatic class declared inside a public class Outer and that appropriate constructor forms are defined, an instance of Inner can be constructed in a static method like this: new Inner()

D. An anonymous class instance that implements the interface MyInterface can be constructed and returned from a method like this:

```
1. return new MyInterface(int x) {
2.     int x;
3.     public MyInterface(int x) {
4.       this.x = x;5.        }6. };
```

16. Which of the following are legal enums?

A.
```
enum Animals { LION, TIGER, BEAR }
```

B.
```
enum Animals {
    int age;
    LION, TIGER, BEAR;
}
```

C.
```
enum Animals {
    LION, TIGER, BEAR;
    int weight;
}
```

D.
```
enum Animals {
    LION(450), TIGER(450), BEAR;
    int weight;
    Animals(int w) {
      weight = w;
    }
}
```

E.
```
enum Animals {
    LION(450), TIGER(450), BEAR;
```

```
    int weight;
    Animals() { }
    Animals(int w) {
      weight = w;
    }
  }
```

17. Which of the following are true? (Choose all that apply.)

A. An enum definition should declare that it extends java.lang.Enum.

B. An enum may be subclassed.

C. An enum may contain public method definitions.

D. An enum may contain private data.

18. Which of the following are true? (Choose all that apply.)

A. An enum definition may contain the main() method of an application.

B. You can call an enum's toString() method.

C. You can call an enum's wait() method.

D. You can call an enum's notify() method.

19. Suppose x and y are of type TrafficLightState, which is an enum. What is the best way to test whether x and y refer to the same constant?

A. if (x == y)

B. if (x.equals(y))

C. if (x.toString().equals(y.toString()))

D. if (x.hashCode() == y.hashCode())

20. Which of the following restrictions apply to anonymous inner classes?

A. They must be defined inside a code block.

B. They may only read and write final variables of the enclosing class.

C. They may only call final methods of the enclosing class.

D. They may not call the enclosing class' synchronized methods.

21. Given the following code, which of the following will compile?

enum Spice { NUTMEG, CINNAMON, CORIANDER, ROSEMARY; }

A. Spice sp = Spice.NUTMEG; Object ob = sp;

B. Spice sp = Spice.NUTMEG; Object ob = (Object)sp;

C. Object ob = new Object(); Spice sp = object;

D. Object ob = new Object(); Spice sp = (Spice)object;

22. Which of the following are true?

A. An anonymous inner class may implement at most one interface.

B. An anonymous inner class may implement arbitrarily many interfaces.

C. An anonymous inner class may extend a parent class other than Object.

D. An anonymous inner class that implements one interface may extend a parent class other than Object.

E. An anonymous inner class that implements several interfaces may extend a parent class other than Object.

23. Which methods return an enum constant's name?

A. getName()

B. name()

C. toString()

D. nameString()

E. getNameString()

Solutions:

1. Answer: C

The default constructor is provided by the compiler.

2. Answer: B

A is incorrect because: the variable on the left had side os of the wrong type (**MyOuterClass.MyNestedClass** is required).

B is correct – the syntax to create a static nested class is:

MyTopLevel.StaticNested sn = new MyTopLevel.StaticNested();

C is incorrect because the syntax on the right hand side of the expression is incorrect. For static nested classes you need to use the outclass class name as in B.

D is incorrect because MyNestedClass is static and cannot belong to a specific instance of the outer class. This might be a little confusing as you can usually access static members (e.g. variables) using a reference of the object e.g. objectname.staticvariable or using the class name followed by the static variable e.g. Classname.staticvariable.

However the code given in D will generate a compiler error. Static nested classes do not have a reference to thye enclosing instance. You need to use the syntax in B.

3. Answer: C

The compiler would place a statement super() in the beginning of constructor B(). However, A()

does not exist. The compiler will not provide it because class A already has a constructor.

4. Answer: D

A, B, C, and E are incorrect statements about an interface.

5. Answer: B

A static method cannot access non-static members of the class.

6. Answer: B and C

The constructor in B would be called from the first and third constructor SubClass(...) in the code, and the constructor in C would be called from the second constructor SubClass(...) in the code.

7. Answer: A

Because the MyClass class already has a constructor, the compiler would not provide a default constructor.

8. Answer: D

The super(j) statement should be the first line in the constructor SubClass(int j).

9. Answer: B and D

A class can inherit only from one class, but a class may inherit from more than one interface. A class inherits from an interface by using the keyword implements and it inherits from another class by using the keyword extends. An interface can inherit from more than one interface by using the keyword extends.

10. Answer: B and D

The list of constants (names) of an enum must come first and must be terminated with a semi-colon (;). It's okay to have a constructor in an enum.

11. Answer: A

A is the correct answer because the variables x and y declared in lines 2 and 3 are different from variables x and y declared in lines 5 and 6.

12. Answer: B,C.

Because the class has explicit constructors defined, the default constructor is suppressed, so A is not possible. B and C have argument lists that match the constructors defined at lines 2 and 4 respectively, and so they are correct constructions. D has three integer arguments, but there are no constructors that take three arguments of any kind in the Test class, so D is incorrect. Finally, E is a

syntax used for construction of inner classes and is therefore wrong.

13. Answer:

A, C. The constructor at lines 2 and 3 includes no explicit call to either this() or super(), which means that the compiler will generate a call to the no-args superclass constructor, as in A. The explicit call to super() at line 5 requires that the Base class must have a constructor as in C. This requirement has two consequences. First, C must be one of the required constructors and therefore one of the answers. Second, the Base class must have at least that constructor defined explicitly, so the default constructor is not generated but must be added explicitly. Therefore the constructor of A is also required and must be a correct answer. At no point in the Test class is there a call to either a superclass constructor with one or three arguments, so B and D need not explicitly exist.

14 .Answer A, B, C, E.

Because Inner is not a static inner class, it has a reference to an enclosing object, and all the variables of that object are accessible. Therefore A and B are correct, despite the fact that b is marked private. Variables in the enclosing method are accessible only if those variables are marked final, so the method argument c is correct, but the variable d is not. Finally, the parameter e is of course accessible, because it is a parameter to the method containing line 8.

15. Answer A.

Construction of a normal (that is, a named and nonstatic) inner class requires an instance of the enclosing class. Often this enclosing instance is provided via the implied this reference, but an explicit reference can be used in front of the new operator, as shown in A. Anonymous inner classes can be instantiated only at the same point they are declared, so B is illegal. C is illegal because Inner is a nonstatic inner class, and so it requires a reference to an enclosing instance when it is constructed. D is illegal because it attempts to use arguments to the constructor of an anonymous inner class that implements an interface.

16. Answer C, E.

A is illegal because the list of names must be terminated by a semicolon. B is illegal because the list of names must be the first element in the enum body. C is a legal enum that contains, in addition to its name list, a variable. D is illegal because the declaration of Bear requires the existence of a no-args constructor. E fixes the bug in D by adding a no-args constructor.

17. Answer: C, D.

Enums may not extend or be extended. They may contain methods and data just like ordinary classes.

18. Answer: A, B, C, D.

Enums may contain public static void main() methods and may serve as application main classes. Enums inherit from Object, so they have toString(), wait(), and notify() methods.

19.Answer: A.

It is never possible to have two instances of an enum that represent the same value. So the == operator is reliable, and it's faster than any method call.

20. Answer A.

An anonymous inner class must appear inside a block of code. There are no restrictions preventing an anonymous inner class from accessing the enclosing class' non-final data or methods or calling the enclosing class' synchronized methods.

21. Answer A, B, D.

Enums may be converted to Object, just like other objects. So A and B are legal, though the cast in B is not necessary. Assigning an Object reference to an enum requires a cast, so C is illegal, but D is legal.

22. Answer: A, C.

An anonymous inner class may either implement a single interface or extend a parent class other than Object, but not both.

23. Answer: B,C.

Both name() and toString() return a constant's name. name() is final, but toString() can be overridden.

Coding Questions:

Methods

- Write code that contains the following:
  - A main method inside a public class called Ex5
  - Another class called TestClass that contains a method that takes three integer values and multiplies them together and returns the value as an int
  - Call the method you have written in the TestClass class from the main method passing in threee integers (you can choose the values)
    - hint: you will need to create an instance of the TestClass so that you can call the method you have created

- Print out the value returned to the main method
- Compile and Run the program

```
1. public class Ex5{
2.     public static void main(String [] args) {
3.         TestClass tc = new TestClass();
4.         System.out.println(tc.mutliply_three_ints(1,2,3));
5.        }
 6.  }
7. class TestClass {
8.         int mutliply_three_ints(int a, int b, int c)
9.         {
10.                 return a*b*c;
11.        }
12.}
```


Static Methods, Variables and Blocks

- Add a static method to your Ex5 class – that has the same functionality as the method you created in TestClass and call it from your main method – again print out the value it returns preceded with the string "static method output:"
- Compile and Run the program
- Add a static block after your static method in Ex5 and print out "this is a static block"
- Compile and Run the program (notice the order of print out statements – the static block was executed first!!!!)

```
1. public class Ex5{
2.     public static void main(String [] args) {
3.         TestClass tc = new TestClass();
4.         System.out.println(tc.mutliply_three_ints(1,2,3));
5.         System.out.println(st_mutliply_three_ints(2,2,3));
6.        }


7.         static int st_mutliply_three_ints(int a, int b, int c)
8.         {
9.                 return a*b*c;
10.        }
11.        static{
12.        System.out.println("This is a static block");
13.        }
14.  }
15. class TestClass {
16.        int mutliply_three_ints(int a, int b, int c)
17.        {
18.                return a*b*c;
19.        }
20.}
```

- Compile, Run and Understand the following code:

```
1. class StaticExample {
2.     static int staticCounter=0;
3.     int counter=0;
4.     StaticExample() {
5.         staticCounter++;
6.         counter++;
7.     }
8. }
9. class RunStaticExample {
10.     public static void main(String[] args) {
11.         StaticExample se1 = new StaticExample();
12.         StaticExample se2 = new StaticExample();
13.         System.out.println("Value of staticCounter for se1: " +  se1.staticCounter);
14.         System.out.println("Value of staticCounter for se2: " +  se2.staticCounter);
15.         System.out.println("Value of counter for se1: " + se1.counter);
16.         System.out.println("Value of counter for se2: " + se2.counter);
17.         StaticExample.staticCounter = 100;
18.         System.out.println("Value of staticCounter for se1: " + se1.staticCounter);
19.     System.out.println("Value of staticCounter for se2: " + se2.staticCounter);
20.     }
21.     }
```

Methods with variable number of parameters

- rewrite the method in the class TestClass such that it can handle variable numbers of parameters

- Compile and Run the program

Program Extract:

```
1. int mutliply_three_ints(int... va)
2.     {
3.         int x=1;
4.         for (int y: va){
5.             x=x*y;
6.         }
7.         return x;
8.     }
```

Nested Classes

- Compile, run and understand the following code:

```
1. class TestNested {
2.   public static void main(String[] args) {
3.     String ext = "From external class";
```

```
4.    MyTopLevel mt = new MyTopLevel();
5.    mt.createNested();
6.    MyTopLevel.MyInner inner = mt.new MyInner();
7.    inner.accessInner(ext);
8.  }
9. }
10. class MyTopLevel{
11.   private String top = "From Top level class";
12.   MyInner minn = new MyInner();
13.   public void createNested() {
14.     minn.accessInner(top);
15.   }
16.   class MyInner {
17.     public void accessInner(String st) {
18.     System.out.println(st);
19.     }
20.   }
21. }
```

- Create a nested class inside the TestClass you created earlier. Add a method to the inner class that prints out the message ”This is an inner class method being called”. Try to call the inner class method from the main method in the Ex5 class.

```
1.  public class Ex5{
2.      public static void main(String [] args) {
3.          TestClass tc = new TestClass();
4.          System.out.println(tc.mutliply_three_ints(1,2,3));
5.          System.out.println(st_mutliply_three_ints(2,2,3));
6.          TestClass.InnerClass ic = tc.new InnerClass();
7.          ic.inner_method();
8.      }
9.
10.     static int st_mutliply_three_ints(int a, int b, int c)
11.     {
12.             return a*b*c;
13.     }
14.
15.     static{
16.     System.out.println("This is a static block");
17.     }
18.
19. }
```

```
20.
21. class TestClass {
22.         int mutliply_three_ints(int... va)
23.         {
24.                 int x=1;
25.
26.                 for (int y: va){
27.
28.                         x=x*y;
29.                 }
30.
31.                 return x;
32.         }
33.         class InnerClass{
34.         void inner_method()
35.         {
36.                 System.out.println("This is an inner class being called");
37.         }
38.         }
39. }
40.
```


Classes inside Methods

- Compile, Run and understand the following code:

```
1. public class MOuter {
2.   private int m = (int)(Math.random() * 100);
3.   public static void main(String args[]) {
4.     MOuter that = new MOuter();
5.     that.go((int)(Math.random() * 100),
6.      (int)(Math.random() * 100));
7.   } // end of main()
8.
9.   public void go(int x, final int y) {
10.    int a = x + y;
11.    final int b = x - y;
12.    class MInner {
13.      public void method() {
14.        System.out.println("m is " + m);
15.        // System.out.println("x is"+x);
16.        System.out.println("y is " + y);
```

```
17.      // System.out.println("a is"+a);
18.      System.out.println("b is " + b);
19.    } // end of method()
20.  } // end of MInner class
21.
22.  MInner that = new MInner();
23.  that.method();
24. } // end of go()
25. } // end of MOuter class
```

- Uncomment out the red lines – try to recompile. What happens? Why?

Enums

- Compile, Run and Understand the following code (to run the code type:

  `java EnumTest charm)`

- Try to tun the code with the different arguments (i.e. up,charm,down, strange, thruth and beauty)

```
1. public class EnumTest {
2.   public static void main(String[] args) {
3.     int nargs=args.length;
4.     if(nargs < 1){
5.       System.out.println("There must be an argument in the command: UP, DOWN,
STRANGE, CHARM, TRUTH, or BEAUTY");
6.       System.out.println("Example: java EnumTest BEAUTY");
7.       System.exit(0);
8.     }else {
9.       Quark q = Enum.valueOf(Quark.class, args[0].toUpperCase());
10.     char symbol = q.getSymbol();
11.     double charge = q.getCharge();
12.     System.out.println("The electric charge for quark " + symbol + ": " + charge);
13.     System.out.println("The name of the quark: " + q.name());
14.   } // end of else
15. } // end of main()
16. } // end of class EnumTest
17. enum Quark {
18.   UP('u', 2.0/3.0),
19.    DOWN('d', -1.0/3.0),
20.    CHARM('c', 2.0/3.0),
21.    STRANGE('s', -1.0/3.0),
22.    TRUTH('t', 2.0/3.0),
23.    BEAUTY('b', -1.0/3.0);
24.   private final char symbol;
25.   private final double charge;
26.   Quark(char symbol, double charge){ // constructor
27.     this.symbol = symbol;
```

```
28.     this.charge = charge;
29.   }
30.   public char getSymbol(){
31.     return symbol;
32.   }
33.   public double getCharge(){
34.     return charge;
35.   }
36. }
```

Writing and Invoking Constructors

- Compile Run and Understand the output of the following code:

```
1. class TestConstructors {
2.    public static void main(String[] args) {
3.      new MySubSubClass();
4.    }
5.
6. }
7. // start of class MySuperClass
8. class MySuperClass {
9.    int superVar = 10;
10.  MySuperClass(){
11.    System.out.println("superVar: " + superVar);
12.  }
13.  MySuperClass(String message) {
14.    System.out.println(message + ": " + superVar);
15.  }
16. } // end of class MySuperClass
17. // Class MySubClass inherits from MySuperClass
18. class MySubClass extends MySuperClass {
19.    int subVar = 20;
20.    MySubClass() {
21.      super("non default super called");
22.      System.out.println("subVar: " + subVar);
23.    }
24. } // end of class MySubClass
25. // Class MySubSubClass inherits from MySubClass
26. class MySubSubClass extends MySubClass {
27.    int subSubVar = 30;
28.    MySubSubClass() {
29.      this("A non-default constructor of MySubSubClass");
```

```
30.      System.out.println("subSubVar: " + subSubVar);
31.    }
32.    MySubSubClass(String message){
33.      System.out.println(message);
34.    }
35. } // end of class MySubSubClass
```

Swap lines 29 and 30 recompile and run. What is the effect? Why?

Exercises and Solutions – Java Language Fundamentals

- Review Questions (Java Language Fundamentals)
- Coding Questions
  - Static Imports
  - Modifiers
  - Passing Arguments

Review Questions :

1. Which of the following declarations will result in compiler error? (Choose all that apply.)

A. abstract final class MyClass {};
B. abstract int i;
C. default class MyClass {};
D. native myMethod();

2. Which of the following statements is false?

A. An abstract class must have at least one abstract method.
B. An abstract class cannot have a finalize() method.
C. A final class cannot have abstract methods.
D. A top-level class cannot be declared private.

3. Assume that a variable exists in a class and that the variable must not be copied into a file when the object corresponding to this class is serialized. What modifier should be used in the declaration of this variable?

A. private
B. protected
C. public
D. transient
E. native

4. Which of the following statements is true about the static modifier?

A. A static variable cannot change its value.
B. A static method cannot be overridden to be non-static.
C. A static method is often written in a non-Java language and exists outside of JVM.
D. The static code lies outside of any class.

5. Consider the following code fragment:

```
1. class MySuperClass {
2. public void message() {
3. System.out.println("From the super class!");
4. }
5. }
6. public class MySubClass extends MySuperClass {
7. void message() {
8. System.out.println("From the subclass!");
9. }
10. public static void main(String args[]) {
11. MySubClass mysub = new MySubClass();
12. mysub.message();
13. }
14. }
```

Which of the following statement is true about this code?

A. The code would compile and execute, and generate the output: From the subclass!.
B. The code would compile and execute, and generate the output: From the super class!.
C. Line 7 would generate a compiler error.
D. Line 11 would generate a compiler error.

6. Consider the following code fragment:

```
1. class MySuperClass {
2. static void message() {
3. System.out.println("From the super class!");
4. }
5. }
6. public class MySubClass extends MySuperClass {
7. void message() {
8. System.out.println("From the subclass!");
9. }
10. public static void main(String args[]) {
11. MySubClass mysub = new MySubClass();
12. mysub.message();
13. }
14. }
```

Which of the following modifiers placed in the beginning of line 7 will make the code compile and execute without error?

A. static
B. public
C. protected
D. transient

7. Which of the following statements is true?

A. The final variable may only be used with a variable or a method.
B. The final variable may not be copied to a file during object serialization.
C. The final method may not be overridden.
D. The class that has a final method may not be extended.

8. Consider the following code fragment:

```
1. class MyClass {
2. public void message (int i) {
3. public int j= i;
4. System.out.println("Value of j: " + j);
5. }
6. public static void main(String[] args) {
7. MyClass ma = new MyClass();
8. ma.message(15);
9. }
10. }
```

Which of the following statements is true about this code?

A. The code will compile and execute fine, and the output will be Value of j: 15.
B. Line 2 will generate a compiler error.
C. The code will compile but give an error at execution time.

Consider the following code fragment for questions 9 and 10:

```
package robots;
public class FunnyRobot {
protected void dance () {
System.out.println("The funny robot is dancing!");
}
void shyAway () {
System.out.println("The funny robot is shying away!");
}
private void freeze () {
System.out.println("The robot has come to a stop!");
}
}
```

9. Consider the following code fragment:

```
1. package RobotDrivers ;
2. import robots.*;
3. public class RobotPlayer extends FunnyRobot {
4. static int i =5;
5. public static void main(String[] args){
6. i = 6;
7. RobotPlayer rp = new RobotPlayer();
8. rp.dance();
9. }
10. }
```

Which of the following statements is true about this code fragment?

A. The code will compile and execute correctly, and generate the output: The funny robot is dancing!.
B. There would be a compiler error at line 7 because the method dance() is protected and the classes RobotPlayer and FunnyRobot are in different packages.
C. There would be a compiler error at line 6.
D. The code will compile, but will generate a runtime exception.

10. Consider the following code fragment:

```
1. package RobotDrivers ;
2. import robots.*;
3. public class RobotPlayer{
4. static int i =5;
5. public static void main(String[] args){
6. i = 6;
7. FunnyRobot fr = new FunnyRobot();
8. fr.dance();
9. }
10. }
```

Which of the following statements is true about this code fragment?

A. The code will compile and execute correctly, and generate the output: The funny robot is dancing!.
B. There would be a compiler error at line 7 because the method dance() is protected and the classes RobotPlayer and FunnyRobot are in different packages.
C. There would be a compiler error at line 6.
D. The code will compile, but will generate a runtime exception.

11. Consider this code:

```
1. class CodeWalkThree {
2. public static void main(String [] args) {
3. CodeWalkThree cw = new CodeWalkThree();
4. CodeWalkThree cw2 = new CodeWalkThree();
5. System.out.print(cw == cw2);
6. cw2 = operate(cw,cw2);
7. System.out.print(" " + (cw == cw2));
8. }
9. static CodeWalkThree operate(CodeWalkThree cw1, CodeWalkThree cw2) {
10. CodeWalkThree cw3 = cw1;
11. cw1 = cw2;
12. return cw3;
13. }
14. }
```

What is the result?

A. false false
B. true true
C. false true
D. true false
E. Compilation fails.

12. If all three top-level elements occur in a source file, they must appear in which order?

A. Imports, package declarations, classes/interfaces/enums
B. Classes/interfaces/enums, imports, package declarations
C. Package declaration must come first; order for imports and class/interfaces/enum definitions is not significant
D. Package declaration, imports, class/interface/enum definitions.
E. Imports must come first; order for package declarations and class/interface/enum definitions is not significant

13. Consider the following application:

```
1. class Q13 {
2. public static void main(String args[]) {
3. Holder h = new Holder();
4. h.held = 100;
5. h.bump(h);
6. System.out.println(h.held);
7. }
8. }
9.
10. class Holder {
11. public int held;
12. public void bump(Holder theHolder) {
13. theHolder.held++; }
14. }
15. }
```

What value is printed out at line 6?
A. 0
B. 1
C. 100
D. 101

14. Consider the following application:

```
1. class Q14 {
2. public static void main(String args[]) {
3. double d = 12.3;
4. Decrementer dec = new Decrementer();
5. dec.decrement(d);
6. System.out.println(d);
7. }
8. }
9.
10. class Decrementer {
11. public void decrement(double decMe) {
12. decMe = decMe - 1.0;
13. }
14. }
```

What value is printed out at line 6?
A. 0.0
B. 1.0
C. 12.3
D. 11.3

15. How can you force garbage collection of an object?

A. Garbage collection cannot be forced.
B. Call System.gc().
C. Call System.gc(), passing in a reference to the object to be garbage-collected.
D. Call Runtime.gc().
E. Set all references to the object to new values (null, for example).

16. Suppose a source file contains a large number of import statements. How do the imports affect the time required to compile the source file?

A. Compilation takes no additional time.
B. Compilation takes slightly more time.
C. Compilation takes significantly more time.

17. Suppose a source file contains a large number of import statements and one class definition. How do the imports affect the time required to load the class?

A. Class loading takes no additional time.
B. Class loading takes slightly more time.
C. Class loading takes significantly more time

18. Which of the following are legal import statements?

A. import java.util.Vector;
B. static import java.util.Vector.*;
C. import static java.util.Vector.*;
D. import java.util.Vector static;

19. Which of the following may be statically imported? (Choose all that apply.)

A. Package names
B. Static method names
C. Static field names
D. Method-local variable names

20. Which of the following are true? (Choose all that apply.)

A. Primitives are passed by reference.
B. Primitives are passed by value.
C. References are passed by reference.
D. References are passed by value.

21. Which of the following declarations are illegal? (Choose all that apply.)

A. default String s;
B. transient int i = 41;
C. public final static native int w();
D. abstract double d;
E. abstract final double hyperbolicCosine();

22. Which of the following statements is true?

A. An abstract class may not have any final methods.
B. A final class may not have any abstract methods.

23. What is the minimal modification that will make this code compile correctly?

```
1. final class Aaa
2. {
3. int xxx;
4. void yyy() { xxx = 1; }
5. }
6.
7.
8. class Bbb extends Aaa
9. {
10. final Aaa finalref = new Aaa();
11.
12. final void yyy()
13. {
14. System.out.println("In method yyy()");
15. finalref.xxx = 12345;
16. }
17. }
```

A. On line 1, remove the final modifier.
B. On line 10, remove the final modifier.
C. Remove line 15.
D. On lines 1 and 10, remove the final modifier.
E. The code will compile as is. No modification is needed.

24. Which of the following statements is true?

A. Transient methods may not be overridden.
B. Transient methods must be overridden.
C. Transient classes may not be serialized.
D. Transient variables must be static.
E. Transient variables are not serialized.

25. Which statement is true about this application?

```
1. class StaticStuff
2 {
3. static int x = 10;
4.
5. static { x += 5; }
6.
7. public static void main(String args[])
8. {
9. System.out.println("x = " + x);
10. }
11.
12. static {x /= 5; }
13. }
```

A. Lines 5 and 12 will not compile because the method names and return types are missing.
B. Line 12 will not compile because you can only have one static initializer.
C. The code compiles and execution produces the output x = 10.
D. The code compiles and execution produces the output x = 15.
E. The code compiles and execution produces the output x = 3.

26. Which statement is true about this code?

```
1. class HasStatic
2. {
3. private static int x = 100;
4.
5. public static void main(String args[])
6. {
7. HasStatic hs1 = new HasStatic();
8. hs1.x++;
9. HasStatic hs2 = new HasStatic();
10. hs2.x++;
11. hs1 = new HasStatic();
12. hs1.x++;
13. HasStatic.x++;
14. System.out.println("x = " + x);
15. }
16. }
```

A. Line 8 will not compile because it is a static reference to a private variable.
B. Line 13 will not compile because it is a static reference to a private variable.
C. The program compiles and the output is x = 102.
D. The program compiles and the output is x = 103.
E. The program compiles and the output is x = 104.

27. Given the following code, and making no other changes, which combination of access modifiers (public, protected, or private) can legally be placed before aMethod() on line 3 and be placed before aMethod() on line 8?

```
1. class SuperDuper
2. {
3. void aMethod() { }
4. }
5.
6. class Sub extends SuperDuper
7. {
8. void aMethod() { }
9. }
```

A. line 3: public; line 8: private
B. line 3: protected; line 8: private
C. line 3: default; line 8: private
D. line 3: private; line 8: protected
E. line 3: public; line 8: protected

28. Which modifier or modifiers should be used to denote a variable that should not be written out as part of its class's persistent state? (Choose the shortest possible answer.)

A. private
B. protected
C. private protected
D. transient
E. volatile

29. This question concerns the following class definition:

```
1. package abcde;
2.
3. public class Bird {
4. protected static int referenceCount = 0;
5. public Bird() { referenceCount++; }
6. protected void fly() { /* Flap wings, etc. */ }
7. static int getRefCount() { return referenceCount; }
8. }
```

Which statement is true about class Bird and the following class Parrot?

```
1. package abcde;
2.
3. class Parrot extends abcde.Bird {
4. public void fly() {
5. /* Parrot-specific flight code. */
6. }
7. public int getRefCount() {
8. return referenceCount;
9. }
10. }
```

A. Compilation of Parrot.java fails at line 4 because method fly() is protected in the superclass, and classes Bird and Parrot are in the same package.
B. Compilation of Parrot.java fails at line 4 because method fly() is protected in the superclass and public in the subclass, and methods may not be overridden to be more public.
C. Compilation of Parrot.java fails at line 7 because method getRefCount() is static in the superclass, and static methods may not be overridden to be nonstatic.
D. Compilation of Parrot.java succeeds, but a runtime exception is thrown if method fly() is ever called on an instance of class Parrot.
E. Compilation of Parrot.java succeeds, but a runtime exception is thrown if method getRefCount() is ever called on an instance of class Parrot.

30. This question concerns the following class definition:

```
1. package abcde;
2.
3. public class Bird {
4. protected static int referenceCount = 0;
5. public Bird() { referenceCount++; }
6. protected void fly() { /* Flap wings, etc. */ }
7. static int getRefCount() { return referenceCount; }
8. }
```

Which statement is true about class Bird and the following class Nightingale?

```
1. package singers;
2.
3. class Nightingale extends abcde.Bird {
4. Nightingale() { referenceCount++; }
5.
6. public static void main(String args[]) {
7. System.out.print("Before: " + referenceCount);
8. Nightingale florence = new Nightingale();
9. System.out.println(" After: " + referenceCount);
10. florence.fly();
11. }
12. }
```

A. The program will compile and execute. The output will be Before: 0 After: 2.
B. The program will compile and execute. The output will be Before: 0 After: 1.
C. Compilation of Nightingale will fail at line 4 because static members cannot be overridden.
D. Compilation of Nightingale will fail at line 10 because method fly() is protected in the superclass.
E. Compilation of Nightingale will succeed, but an exception will be thrown at line 10, because method fly() is protected in the superclass.

31. Suppose class Supe, in package packagea, has a method called doSomething(). Suppose class Subby, in package packageb, overrides doSomething(). What access modes may Subby's version of the method have? (Choose all that apply.)

A. public
B. protected
C. Default
D. private

32. Which of the following statements are true?

A. An abstract class may be instantiated.
B. An abstract class must contain at least one abstract method.
C. An abstract class must contain at least one abstract data field.
D. An abstract class must be overridden.
E. An abstract class must declare that it implements an interface.
F. None of the above.

33. Suppose interface Inty defines five methods. Suppose class Classy declares that it implements Inty but does not provide implementations for any of the five interface methods. Which is/are true?

A. The class will not compile.
B. The class will compile if it is declared public.
C. The class will compile if it is declared abstract.
D. The class may not be instantiated.

34. Which of the following may be declared final? (Choose all that apply.)

A. Classes
B. Data
C. Methods

35. Which of the following may follow the static keyword? (Choose all that apply.)

A. Class definitions
B. Data
C. Methods
D. Code blocks enclosed in curly brackets

36. Suppose class A has a method called doSomething(), with default access. Suppose class B extends A and overrides doSomething(). Which access modes may apply to B's version of doSomething()? (Choose all that apply.)

A. public
B. private
C. protected
D. Default

37. True or false: If class Y extends class X, the two classes are in different packages, and class X has a protected method called abby(), then any instance of Y may call the abby() method of any other instance of Y.

A. True
B. False

38. Which of the following statements are true?

A. A final class must be instantiated.
B. A final class must contain at least one final method.
C. A final class must contain at least one final data field.
D. A final class may not be extended.
E. None of the above.

39. Which of the following statements are true?

A. A final class must be instantiated.
B. A final class may only contain final methods.
C. A final class may not contain non-final data fields.
D. A final class may not be extended.
E. None of the above.

40. What does the following code print?

```
public class A
{
static int x;
public static void main(String[] args) {
A that1 = new A();
A that2 = new A();
that1.x = 5;
that2.x = 1000;
x = -1;
System.out.println(x);
}
}
```

A. 0
B. 5
C. 1000
D. -1

Solutions – Review Questions:

1. Answer: A, B, and C
A class cannot be abstract and final at the same time, a variable cannot be declared abstract, and default is not a keyword (modifier) in Java.

2. Answer: A and B
An abstract class does not have to have any abstract method and cannot have a finalize() method.

3. Answer: D
A variable declared transient is not saved during serialization.

4. Answer: B
A, C, and D are false statements about the static modifier.

5. Answer: C
A method cannot be overridden to be less public.

6. Answer: A
A call is being made to the message() method from the static method main(…). So, the message() method must be static.

7. Answer: C
A, B, and D are false statements about the final modifier.

8. Answer: B
The method variables cannot have access modifiers.

9. Answer: A
There will be no compiler error at line 6 because a static method can access another static variable of the class. There will be no compiler error at line 7 either because the default constructor is provided by the compiler.

10. Answer: B
RobotPlayer is in a different package from FunnyRobot and is not a subclass of FunnyRobot and thus has no access to protected members of FunnyRobot.

11. Answer: C
C is the correct answer because the object references cw and cw2 point to different objects before the method call, and they point to the same object after the method call.

12. Answer: D. Package declaration must come first, followed by imports, followed by class/interface/enum definitions.

13 Answer:. D. A holder is constructed on line 3. A reference to that holder is passed into method bump() on line 5. Within the method call, the holder's held variable is bumped from 100 to 101.

14. Answer: C. The decrement() method is passed a copy of the argument d; the copy gets decremented, but the original is untouched.

15. Answer: A. Garbage collection cannot be forced. Calling System.gc() or Runtime.gc() is not 100 percent reliable, because the garbage-collection thread might defer to a thread of higher priority; thus options B and D are incorrect. Option C is incorrect because the two gc() methods do not take arguments; in fact, if you still have a reference to pass into any method, the object is not yet eligible to be collected. Option E will make the object eligible for collection the next time the garbage collector runs.

16. Answer: B . Importing slightly increases compilation time.

17. Answer: A.. Importing is strictly a compile-time function. It has no effect on class loading or on any other run-time function.

18. Answer: A, C. The import keyword may optionally be followed by the static keyword.

19. Answer: B, C. You may statically import method and field names.

20. Answer: B, D. In Java, all arguments are passed by value.

21 Answer:. A, D, E. A is illegal because "default" is not a keyword. B is a legal transient declaration. C is strange but legal. D is illegal because only methods and classes may be abstract. E is illegal because abstract and final are contradictory.

22. Answer: B. Any class with abstract methods must itself be abstract, and a class may not be both abstract and final. Statement A says that an abstract class may not have final methods, but there is nothing wrong with this. The abstract class will eventually be subclassed, and the subclass must avoid overriding the parent's final methods. Any other methods can be freely overridden.

23. Answer: A. The code will not compile because on line 1, class Aaa is declared final and may not be subclassed. Lines 10 and 15 are fine. The instance variable finalref is final, so it may not be modified; it can reference only the object created on line 10. However, the data within that object is not final, so nothing is wrong with line 15.

24. Answer: E. A, B, and C don't mean anything because only variables may be transient, not methods or classes. D is false because transient variables need not be static, and in fact they very rarely are.. E is a good one-sentence definition of transient.

25. Answer: E. Multiple static initializers (lines 5 and 12) are legal. All static initializer code is executed at class-load time, so before main() is ever run, the value of x is initialized to 10 (line 3), then bumped to 15 (line 5), and then divided by 5 (line 12).

26. Answer: E. The program compiles fine; the "static reference to a private variable" stuff in answers A and B is nonsense. The static variable x gets incremented four times, on lines 8, 10, 12, and 13.

27. Answer: D. The basic principle is that a method may not be overridden to be more private. All choices except D make the access of the overriding method more private.

28. Answer: D. A and B are access modifiers. C is an illegal combination of two access modifiers. E ("volatile") is used for certain multi-threaded situations, and is not covered in the Exam.

29. Answer: C. Static methods may not be overridden to be nonstatic. B is incorrect because it states the case backward: methods can be overridden to be more public, not more private. Answers A, D, and E make no sense.

30. Answer: A. There is nothing wrong with Nightingale. The static referenceCount is bumped twice: once on line 4 of Nightingale and once on line 5 of Bird. (The no-argument constructor of the superclass is always implicitly called at the beginning of a class's constructor, unless a different superclass constructor is requested. This has nothing to do with modifiers; Because referenceCount is bumped twice and not just once, answer B is wrong.C says that statics cannot be overridden, but no static method is being overridden on line 4; all that is happening is an increment of an inherited static variable. D is wrong because protected is precisely the access modifier you want Bird.fly() to have: you are calling Bird.fly() from a subclass in a different package. Answer E is ridiculous, but it uses credible terminology.

31. Answer: A, B. Since the method in the superclass is overridden in a different package, the superclass version must be public or protected. (Default methods may be overridden only if the subclass is in the same package as the superclass; private methods may not be overridden at all.) An overriding method's access mode must be the same as, or more open than, the superclass version's access mode.

32. Answer: F. A is false because the compiler forbids construction of an abstract class. B is false because abstract classes need not contain any abstract methods (though if a class *does* contain any abstract methods it *must* be abstract). C is false because there is no such thing as an abstract data field. D is false because, when you really think about it, it doesn't make any sense; the compiler compiles classes one at a time and has no interest in whether or not a class is overridden. E is false because there is no compiler requirement that an abstract class must declare that it implements any interface.

33. Answer: C, D. If a class does not provide implementations for all methods of all interfaces that the class declares it implements, that class must be declared abstract. Abstract classes may not be instantiated.

34. Answer: A, B, C. Classes, data, and methods can be declared final. Variables cannot.

35. Answer: B, C, D. Classes may not be static. Data and methods may be static, and often they are. When the static keyword is followed by a code block, the block is a static initializer and is executed when the class is loaded.

36. Answer: A, C, D. An overriding method's access mode must be the same as, or more open than, the superclass version's access mode.

37. Answer: B. An object that inherits a protected method from a superclass in a different package may call that method on itself but not on other instances of the same class.

38. Answer: D. There is only one restriction on a final class: A final class may not be extended.

39. Answer: D. There is only one restriction on a final class: A final class may not be extended.

40. Answer: D. Since x is static, there is only one variable, which is shared by all instances along with the class. Thus the last assignment wins.

Coding Questions:

Static Import

- Write a class that performs a static import of PI and E from the java.lang.math package and prints out their values in the main method.
- Compile and Run the program

Solution:

```
import static java.lang.Math.PI;

import static java.lang.Math.E;

class StaticImportTest{

  public static void main(String[] args) {

      System.out.println("Pi:" + PI);

      System.out.println("E:" + E);

  }

}
```

Modifiers

- Write code that contains the following:
  - An abstract class called vehicle that contains two methods
    - An abstract method called drive that returns no value
    - A method called message that prints out a string " This is a abstract parent"
  - A class Car that extends vehicle and that implements the drive method and that prints out the value "Car driving"
  - A class Truck that extends vehicle and that implements the drive method and that prints out the value "Truck driving"
  - A public class called RunVehicle that has
    - a main method which
      - creates and instance of Car and of Truck
      - calls the drive and message methods on each instance
  - Compile and Run the program

Solution:

```
abstract class Vehicle {
   abstract void drive();
   void message() {
      System.out.println("This is an abstract parent.");
   }
}
 class Car extends Vehicle {
   void drive() {
      System.out.println("Car driving.");
   }
 }
 class Truck extends Vehicle {
   void drive() {
      System.out.println("Truck driving.");
   }
 }
 public class RunVehicle {
   public static void main(String[] args) {
      Car car = new Car();
      Truck truck = new Truck();
       car.drive();
       truck.drive();
       truck.message();
   }
}
```

- Examine the following code – why will it not compile?
  - Comment out the necessary lines to compile the code without errors
  - Compile, Run and Understand the code

```
1. class Calculator {
2. final int dime = 10;
3. int count = 0;
4. Calculator (int i) {
5. count = i;
6. }
7. }
8. class RunCalculator {
9. public static void main(String[] args) {
10. final Calculator calc = new Calculator(1);
11. calc = new Calculator(2); // compiler error.
12. calc.count = 2; //ok
13. calc.dime = 11; // compiler error.
14. System.out.println("dime: " + calc.dime);
15. }
16. }
```

Passing Arguments

- Compile, Run and Understand the following code:

```
1. class Student {
2. public static void main (String [] args) {
3. int score = 75;
4. Student st = new Student();
5. st.modifyStudent(score);
6. System.out.println("The original student score: " + score);
7. }
8. void modifyStudent(int i){
9. i = i+10;
10. System.out.println("The modified student score: " + i);
11. }
12.}
```

- Compile, Run and Understand the following code:

```
1. class TestRefVar {
2. public static void main (String [] args) {
3. Student st = new Student("John", 100);
4. System.out.println("The original student info:");
5. System.out.println("name: " + st.getName() + " score: " +
6. st.getScore());
7. TestRefVar tr = new TestRefVar();
8. tr.modifyRef(st);
9. System.out.println("The modified student info in the calling method:");
10. System.out.println("name: " + st.getName() + " score: " + st.getScore());
11. }
12. void modifyRef(Student student){
13. student.setScore(50);
14. student = new Student("Mary", 75);
15. System.out.println("The modified student info in the called method:");
16. System.out.println("name: " + student.getName() + " score: " +
student.getScore());
17. }
18. }
19. class Student {
20. int score;
21. String name;
22. Student(String st, int i){
23. score=i;
24. name=st;
25. }
26. String getName(){
27. return name;
28. }
29. int getScore(){
30. return score;
31. }
32. void setScore(int i){
33. score = i;
34. }
35.}
```

Exercises and Solutions – Object Oriented Programming

- Review Questions (Object Oriented Programming)
- Coding Questions
  - Encapsulation
  - Polymorphism
  - Conversion and Casting of Primitive Types
  - Conversion and Casting of Object Reference Types
  - Overriding and Overloading

Review Questions :

1. Consider the following line of code:

short s = 9L;

What would be the output? (Choose all that apply.)
A. Compiler error
B. Runtime error
C. No error
D. Loss of accuracy

2. Which of the following statements is true?

A. Only primitive data types, and not the object references, can be converted implicitly.
B. Only the object references, and not the primitive data types, can be cast (converted explicitly).
C. Both object references and primitive data types may be converted implicitly and explicitly.
D. Casting primitive data types is checked only at execution time.

3. Consider the following line of code:

short s = 9;

What would be the output of this line of code? (Choose all that apply.)
A. Compiler error
B. Runtime error
C. No error
D. Loss of accuracy

4. Consider the following line of code:

byte b = 335;

What would be the output of this line of code?
A. Compiler error
B. Runtime error
C. No error
D. Loss of accuracy

5. If the following code works correctly, what are the possible types of variable c?

```
byte a = 7;
short b = 3;
c = a * ++b;
```

A. short, int, long, float, double
B. short, char, int, float, double
C. byte, short, int, long, float, double
D. int, long, float, double

6. Consider the following code fragment:

```
1. class Student {
2. private int studentId = 0;
3. void setStudentID (int sid) {
4. studentId = sid;
5. System.out.println("Student ID has been set to " + sid);
6. }
7. public static void main(String args[]) {
8. short s = 420;
9. Student st1 = new Student();
10. st1.setStudentID(s);
11. }
12. }
```

Which of the following statements about this code is true?

A. Line 10 will generate a compiler error because the method setStudentID(…) takes an int argument, and not a short argument.
B. The code will compile but will throw an exception at execution time due to line 10.
C. The code will compile, execute, and produce the output Student ID has been set to 420.
D. Line 8 will generate a compiler error.

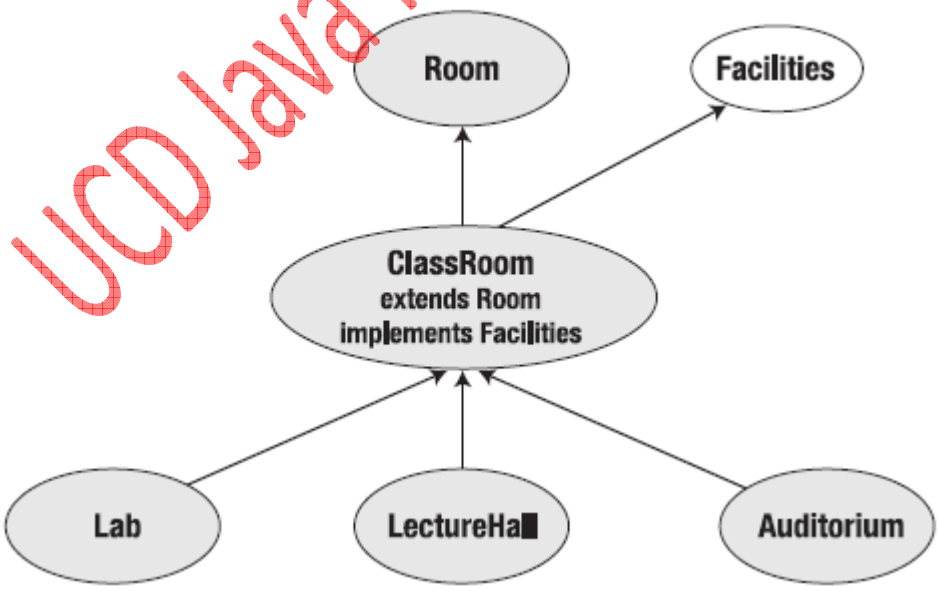
7. Which of the following statements about object reference conversion is true?

A. Object references can never be converted.
B. Object references can never be converted in method calls.
C. Object references can be converted both in method calls and in assignments, but the rules for both are different.
D. Object references can be converted both in method calls and in assignments, and the rules for both are the same.

8. Consider the following code fragment:

```
1. class StudentProb {
2. private int studentId = 0;
3. void setStudentID(int sid) {
4. student_id = sid;
5. System.out.println("Student ID has been set to " + sid);
6. }
7. public static void main(String args[]) {
8. int i = 420;
9. Object ob1;
10. StudentProb st1 = new StudentProb();
11. ob1 = st1;
12. st1.setStudentID(i);
13. }
14. }
```

Which of the following statements is true about this code?
A. A compiler error will occur due to line 9.
B. A compiler error will occur due to line 11.
C. An exception will occur during execution due to line 11.
D. The code will compile, execute, and produce the output Student ID has been set to 420.



For questions 9 and 10, consider the class hierarchy shown above:

9. Consider the following code fragment:

```
1. LectureHall lh = new LectureHall();
2. Auditorium a1;
3. Facilities f1;
4.
5. f1 = lh;
6. a1 = f1;
```

What of the following is the true statement about this code?

A. The code will compile and execute without any error.
B. Line 5 will generate a compiler error because an explicit conversion (cast) is required.
C. Line 6 will generate a compiler error because an explicit conversion (cast) is required to convert Facilities to Auditorium.

10. Consider the following code fragment:

```
1. LectureHall lh = new LectureHall();
2. Auditorium a1;
3. Facilities f1;
4.
5. f1 = lh;
6. a1 = (Auditorium) f1;
```

What of the following is the true statement about this code?

A. The code will compile and execute without any error.
B. Line 5 will generate a compiler error because an explicit conversion (cast) is required.
C. Line 6 will generate a compiler error because an interface cannot be converted to the class that implements the interface.
D. Line 6 will compile fine, but an exception will be thrown during the execution time.

11. Consider the following class:

```
1. class MyClass {
2. public int myMethod (double a, int i) {
3. }
4.
5. }
```

Which of the following methods, if added at line 4 independently, would be valid?

A. public int myMethod(int i, double a) { }
B. public double myMethod(double b, int j){ }
C. public int myMethod(double a, double b, int i){ }
D. public int yourMethod(double a, int i) { }

12. You have been given a design document for implementation in Java. It states:

A room has a table, and a chair. A classroom is a room that has a teacher and students. Assume that the Room *class has already been defined.*

Which of the following data members would be appropriate to include in the class ClassRoom?

A. Room aRoom;
B. Table aTable;
C. Chair aChairbb;
D. Teacher aTeacher;
E. Student aStudent;

13. Which of the following statements are true?

A. Inheritance represents an *is-a* relationship.
B. Inheritance represents a *has-a* relationship.
C. An instance represents an *is-a* relationship.
D. An instance represents a *has-a* relationship.
E. A method represents an *is-a* relationship.

14. A laptop is a specific kind of computer, and it has a network card installed on it. A desktop is also a specific kind of computer. Which of the following statements are true?

A. A laptop *is-a* computer.
B. A computer *is-a* laptop.
C. A laptop *has-a* network card.
D. A computer *is-a* desktop

15. Consider the following classes defined in separate source files:

```
class SuperClass {
SuperClass() {
System.out.print(" I was in Super Class." );
}
public void aMethod (int i) {
System.out.print (" The value of i is " + i );
}
}
class SubClass extends SuperClass {
public void aMethod(int j) {
System.out.print (" The value of j is " + j );
}
}
class Test {
public static void main(String args[]) {
SubClass sub = new SubClass();
sub.aMethod(5);
}
}
```

What output is generated when the class Test is run?

A. The value of i is 5.
B. The value of j is 5.
C. I was in Super Class. The value of i is 5.
D. I was in Super Class. The value of j is 5.

16. Consider this code :

```
1. class CodeWalkFour {
2. public static void main(String[] args){
3. Car c = new Lexus();
4. System.out.print(c.speedUp(30) + " ");
5. Lexus l = new Lexus();
6. System.out.print(l.speedUp(30, 40, 50));
7. }
8. }
9. class Car {
10. private int i=0;
11. int speedUp(int x){
12. return i;
13. }
14. }
15. class Lexus extends Car {
16. private int j = 1;
17. private int k = 2;
18. int speedUp(int y){
19. return j;
20. }
21. int speedUp(int... z){
22. return k;
23. }
24. }
```

What is the result?

A. 30 30 40 50
B. 0 1
C. 0 2
D. 1 2
E. Compilation fails.

17. Consider this class:

```
1. public class Test1 {
2. public float aMethod(float a, float b) {
3. }
4.
5. }
```

Which of the following methods would be legal if added (individually) at line 4? (Choose all that apply.)

A. public int aMethod(int a, int b) { }
B. public float aMethod(float a, float b) { }
C. public float aMethod(float a, float b, int c) throws Exception { }
D. public float aMethod(float c, float d) { }
E. private float aMethod(int a, int b, int c) { }

18. Consider these classes, defined in separate source files:

```
1. public class Test1 {
2. public float aMethod(float a, float b)
3. throws IOException {...
4. }
5. }
```

```
1. public class Test2 extends Test1 {
2.
3. }
```

Which of the following methods would be legal (individually) at line 2 in class Test2? (Choose all that apply.)

A. float aMethod(float a, float b) {...}
B. public int aMethod(int a, int b) throws Exception {...}
C. public float aMethod(float a, float b) throws Exception {...}
D. public float aMethod(float p, float q) {...}

19. You have been given a design document for a veterinary registration system for implementation in Java. It states:

"A pet has an owner, a registration date, and a vaccination-due date. A cat is a pet that has a flag indicating whether it has been neutered, and a textual description of its markings."

Given that the Pet class has already been defined, which of the following fields would be appropriate for inclusion in the Cat class as members? (Choose all that apply.)

A. Pet thePet;
B. Date registered;
C. Date vaccinationDue;
D. Cat theCat;
E. boolean neutered;
F. String markings;

20. You have been given a design document for a veterinary registration system for implementation in Java. It states:

"A pet has an owner, a registration date, and a vaccination-due date. A cat is a pet that has a flag indicating if it has been neutered, and a textual description of its markings."

Given that the Pet class has already been defined and you expect the Cat class to be used freely throughout the application, how would you make the opening declaration of the Cat class, up to but not including the first opening brace? Use only these words and spaces: boolean, Cat, class, Date, extends, Object, Owner, Pet, private, protected, public, String.

A. protected class Cat extends Owner
B. public class Cat extends Object
C. public class Cat extends Pet
D. private class Cat extends Pet

21. Consider the following classes, declared in separate source files:

```
1. public class Base {
2. public void method(int i) {
3. System.out.print("Value is " + i);
4. }
5. }
```

```
1. public class Sub extends Base {
2. public void method(int j) {
3. System.out.print("This value is " + j);
4. }
5. public void method(String s) {
6. System.out.print("I was passed " + s);
7. }
8. public static void main(String args[]) {
9. Base b1 = new Base();
10. Base b2 = new Sub();
11. b1.method(5);
12. b2.method(6);
13. }
14. }
```

What output results when the main method of the class Sub is run?

A. Value is 5Value is 6
B. This value is 5This value is 6
C. Value is 5This value is 6
D. This value is 5Value is 6
E. I was passed 5I was passed 6

22. Consider the following class definition:

```
1. public class Test extends Base {
2. public Test(int j) {
3. }
4. public Test(int j, int k) {
5. super(j, k);
6. }
7. }
```

Which of the following forms of constructor must exist explicitly in the definition of the Base class? Assume Test and Base are in the same package. (Choose all that apply.)

A. Base() { }
B. Base(int j) { }
C. Base(int j, int k) { }
D. Base(int j, int k, int l) { }

23. Which of the following may override a method whose signature is void xyz(float f)?

A. void xyz(float f)
B. public void xyz(float f)
C. private void xyz(float f)
D. public int xyz(float f)
E. private int xyz(float f)

24. Suppose class X contains the following method:

void doSomething(int a, float b) { … }

Which of the following methods may appear in class Y, which extends X?

A. public void doSomething(int a, float b) { … }
B. private void doSomething(int a, float b) { … }
C. public void doSomething(int a, float b)
throws java.io.IOException { … }
D. private void doSomething(int a, float b)
throws java.io.IOException { … }

25. This question involves IOException, AWTException, and EOFException. They are all checked exception types. IOException and AWTException extend Exception, and EOFException extends IOException.

Suppose class X contains the following method:
void doSomething() throws IOException{ … }

Which of the following methods may appear in class Y, which extends X?

A. void doSomething() { … }
B. void doSomething() throws AWTException { … }
C. void doSomething() throws EOFException { … }
D. void doSomething() throws IOException, EOFException { … }

Solutions:

1. Answer: A
A compiler error will occur because 9L is not an int literal. It is a long literal.

2. Answer: C
A, B, and D are false statements about conversion.

3. Answer: C
No error would occur because the conversion rule is relaxed when an int literal value is assigned to a variable of type byte, short, or char, provided the value falls within the range of the target type.

4. Answer: A
The number 335 is not within the range of values that a byte can hold.

5. Answer: D
The result of the calculation on the right side of the assignment is an integer. An integer may be assigned to int, long, float, or double.

6. Answer: C
The short will be converted to int during the method call.

7. Answer: D
A, B, and C are false statements about object reference conversion.

8. Answer: D
Lines 9 and 11 will not produce errors because the conversion rules are being followed.

9. Answer: C
You cannot implicitly convert Facilities to Auditorium because Facilities is not a subinterface of Auditorium.

10. Answer: D
You cannot explicitly convert LectureHall into Auditorium because neither of these classes is a subclass of the other.

11. Answer: A, C, and D
A and C are properly overloaded. D is a different method. B has the same argument types in the same order, so it is not a valid overloading.

12. Answer: D and E
The Classroom class has a *has-a* relationship with the Student class and the Teacher class.

13. Answer: A and D
B, C, and E are incorrect statements about inheritance.

14. Answer: A and C
B and D are incorrect statements about the *is-a* relationship.

15. Answer: D
The no-argument constructor of the SuperClass class is called in the chain of constructor calls.

16. Answer: D
Due to the method call in line 4, the non-vararg version of the speedup(…) method of Lexus is called, whereas due to the call in line 6, the vararg version of the method is called.

17. Answer: A, C, E. In each of these answers, the argument list differs from the original, so the method is an overload. Overloaded methods are effectively independent, and there are no constraints on the accessibility, return type, or exceptions that may be thrown. B would be a legal overriding method, except that it cannot be defined in the same class as the original method; rather, it must be declared in a subclass. D is also an override, because the *types* of its arguments are the same: changing the parameter names is not sufficient to count as overloading.

18. Answer: B, D. A is illegal because it is less accessible than the original method; the fact that it throws no exceptions is perfectly acceptable. B is legal because it overloads the method of the parent class, and as such it is not constrained by any rules governing its return value, accessibility, or argument list. The exception thrown by C is sufficient to make that method illegal. D is legal because the accessibility and return type are identical, and the method is an override because the types of the arguments are identical—remember that the names of the arguments are irrelevant. The absence of an exception list in D is not a problem: An overriding method may legitimately throw fewer exceptions than its original, but it may not throw more.

19. Answer: E, F. The Cat class is a subclass of the Pet class, and as such should extend Pet, rather than contain an instance of Pet. B and C should be members of the Pet class and as such are inherited into the Cat class; therefore, they should not be declared in the Cat class. D would declare a reference to an instance of the Cat class, which is not generally appropriate inside the Cat class (unless, perhaps, you were asked to give the Cat a member that refers to its mother). Finally, the neutered flag and markings descriptions, E and F, are the items called for by the specification; these are correct items.

20. Answer: C. The class should be public, because it is to be used freely throughout the application. The statement "A cat is a pet" tells you that the Cat class should subclass Pet. The other words offered are required for the body of the definitions of either Cat or Pet—for use as member variables—but are not part of the opening declaration.

21. Answer: C. The first message is produced by the Base class when b1.method(5) is called and is therefore Value is 5. Despite the fact that variable b2 is declared as being of the Base class, the behavior that results when method() is invoked upon it is the behavior associated with the class of the actual object, not with the type of the variable. Because the object is of class Sub, not of class Base, the second message is generated by line 3 of class Sub: This value is 6.

22. Answer: A, C. The constructor at lines 2 and 3 includes no explicit call to either this() or super(), which means that the compiler will generate a call to the no-args superclass constructor, as in A. The explicit call to super() at line 5 requires that the Base class must have a constructor as in C. This requirement has two consequences. First, C must be one of the required constructors and therefore one of the answers. Second, the Base class must have at least that constructor defined explicitly, so the default constructor is not generated but must be added explicitly. Therefore the constructor of A is also required and must be a correct answer. At no point in the Test class is there a call to either a superclass constructor with one or three arguments, so B and D need not explicitly exist.

23. Answer: A, B. A uses the original method's signature verbatim, which is legal. B makes the subclass version more accessible, which is legal. C makes the subclass version less accessible, which is not legal. D and E change the return type, which is not legal.

24. Answer: A. A method with default access may be overridden to have default, protected, or public access but not private access, because a method may not be overridden with more restrictive access. An overriding method may not declare that it throws exception types that do not appear in the superclass version.

25. Answer: A, C, D. An overriding method may throw an unlimited number of exception types, provided all types are the same as, or are descended from, the types that appear in the overridden version's declared list of thrown exception types.

Coding Questions:

Encapsulation

- Compile and run the code below
- Add an instance variable "painkiller" of type private String to the class EncapsulateGood
- Create two public methods that allow for setting and getting the value of "painkiller"
- From the main method set the value of "painkiller" to "paracetamol"
- From the main method print out the value in "painkiller"
- Compile and Run the program

```
1. public class TestEncapsulateGood {
2. public static void main(String[] args) {
3. EncapsulateGood eg = new EncapsulateGood();
4. eg.setHeadache(false);
5. System.out.println("Do you have a headache? " + eg.getHeadache());
6. }
7. }

8. class EncapsulateGood {
9. private boolean headache = true;
10. private int doses = 0;
11. public void setHeadache(boolean isHeadache){
12. this.headache = isHeadache;
13. }
14. public boolean getHeadache( ){
15. return headache;
16. }
17. }
```

Solution:

```
public class TestEncapsulateGood {
 public static void main(String[] args) {
   EncapsulateGood eg = new EncapsulateGood();
   eg.setHeadache(false);
   System.out.println("Do you have headache? " + eg.getHeadache());
   eg.setPainkiller("paracetamol");
   System.out.println(eg.getPainkiller());
 }
}
class EncapsulateGood {
  private boolean headache = true;
  private int doses = 0;
  private String painkiller;
  public void setHeadache(boolean isHeadache){
     this.headache = isHeadache;
  }
  public boolean getHeadache( ){
     return headache;
  }
  public void setPainkiller(String painkiller){
     this.painkiller = new String(painkiller);
  }
  public String getPainkiller( ){
     return painkiller;
  }
}
```

Polymorphism

- Write code that contains the following:
  - A class called vehicle that contains two methods
    - A method called drive that returns a string "Driving your Vehicle"
  - A class Car that extends vehicle and that overrides the drive method and that prints out the value "Driving your Car"
  - A class Truck that extends vehicle and that overrides the drive method and that prints out the value "Driving your Truck"
  - A public class called RunVehicle that has
    - a main method which
      - creates a local variable of type Vehicle called "vehicle"
      - assignes an instance of Vehicle to the "vehicle"
      - calls the drive method using "vehicle.drive()"
      - creates an instance of Car and assigns it to "vehicle"
      - again calls the drive method using "vehicle.drive()"
      - creates an instance of Truck and assigns it to "vehicle"
      - again calls the drive method using "vehicle.drive()"
  - Compile and Run the program

Solution:

```
public class RunVehicle {
 public static void main(String [] args) {
  Vehicle vehicle = new Vehicle();
  Car car= new Car();
  Truck truck = new Truck();
  vehicle.drive();
  vehicle=car;
  vehicle.drive();
  vehicle=truck;
  vehicle.drive();
 }
}
class Vehicle {
  public void drive() {
     System.out.println("Driving your vehicle");
  }
}
class Car extends Vehicle {
  public void drive() {
     System.out.println("Driving your car");
  }
}
class Truck extends Vehicle{
  public void drive() {
     System.out.println("Driving your truck");
  }
}
```

Conversion of Primitive Data Types

- Examine the following code – why will it not compile?
- You can use the compiler to help you figure out the answer
- **(Solution see slides 31 and 32)**

```
1 public class ConversionToNarrower{
2. public static void main(String[] args) {
3. int i = 15;
4. short s = 10;
5. s = i;
6. System.out.println("Value of s: " + s );
7. }
8. }
```

- Examine the following code – why will it not compile?
- You can use the compiler to help you figure out the answer
- **(Solution see slide 36 and 37)**

```
1. public class ConversionTest{
2. public static void main(String[] args) {
3. short s = 10;
4. char c = 'a';
5. long l = 16;
6. float f = 1.2f;
7. f = l;
8. c = s;
9. i = f;
10. System.out.println("Value of f: " + f );
11. }
12. }
```

- Examine the following code – why will it not compile?
- You can use the compiler to help you figure out the answer
- **(Solution see slides 41 and 42)**

```
1. public class MethodCallTest{
2. public static void main(String[] args) {
3. int i = 15;
4. long j = 16;
5. byte b = 8;
6. short s = 9;
7. float f = 1.2f;
8. double d = 2.56d;
9. int result1, result2;
10. MethodCallTest mct = new MethodCallTest();
11. result1 = mct.add(f, d);
12. result2 = mct.add(b, s);
13. System.out.println("result2: " + result2 );
14. }
15. public int add(int i, int j) {
16. return (i+j);
17. }
18. }
```

Casting Primitive Data Types

- Examine the code below
- What output do you expect?
- Compile, Run the code – were you surprised by the output? Can you explain it?
- **For a solution see week by week questions – week 9**
- Change the value on line 5 to 200
- Compile, Run the code
- Can you explain the output?

```
1. public class Casting{
2. public static void main(String[] args) {
3. long l = 10;
4. int i = (int) l;
5. short s = 175;
6. byte b = (byte) s;
7. System.out.println("Value of i: " + i );
8. System.out.println("Value of b: " + b);
9. }
10. }
```

Conversion and Casting Object Reference Types

- Try to compile the code below – why is there an error?
- Remove the line that causes the error – recompile and run
- **(Solution – see slides 61 and 62)**

```
1. public class ObjectRefConversion{
2. public static void main(String[] args) {
3. ClassRoom[] crooms = new ClassRoom[10];
4. Room[] rooms;
5. Lab[] labs;
6. for (int i = 0; i<5; i++) {
7. crooms[i] = new ClassRoom();
8. }
9. rooms = crooms;
10. labs = crooms; // compiler error
11. }
12. }
13.
14. interface Facilities {
15. }
16. class Room {
17. }
18. class ClassRoom extends Room implements Facilities{
19. }
20. class Lab extends ClassRoom {
21. }
```

- Write an application that illustrates legal and illegal casts.
- Work with the following class/interface hierarchy:

class Fruit
class Apple extends Fruit
interface Squeezable
class Citrus extends Fruit implements Squeezable
class Orange extends Citrus

- You will have to define the classes and the interface, but the definitions can be empty. Your application should construct one instance of each of the following classes:
    - Object
    - Fruit
    - Apple
    - Citrus
    - Orange
- Try to cast each of these objects to the following types:
    - Fruit
    - Apple
    - Squeezable
    - Citrus
    - Orange

- For each attempted cast, print out a message stating whether the cast succeeded. (A ClassCastException is thrown if the cast failed; if no exception is thrown, the cast succeeded.)
- A fragment of the output of the sample solution looks like this:

Checking casts for FruitFruit: OKApple: NOSqueezable: NOCitrus: NOOrange: NO
Checking casts for AppleFruit: OKApple: OKSqueezable: NOCitrus: NOOrange: NO

Overloading and Overriding

- Understand, compile and run the code below:
- Which methods are overloaded/overridden??
- **Solution:**
    - **The int speedUp(int x) method in the Car class is overridden by the int speedUp(int y) in the class Lexus**
    - **The int speedup method in classs Lexus is overloaded**
- Create a class called Merc that extends Car and that overrides speedup and returns the int value 10
- Create an instance of Merc and call the speedup method – print out the value

```
1. class CodeWalkFour {
2. public static void main(String[] args){
3. Car c = new Lexus();
4. System.out.print(c.speedUp(30) + " ");
5. Lexus l = new Lexus();
6. System.out.print(l.speedUp(30, 40, 50));
7. }
8. }
9. class Car {
10. private int i=0;
11. int speedUp(int x){
12. return i;
13. }
14. }
15. class Lexus extends Car {
16. private int j = 1;
17. private int k = 2;
18. int speedUp(int y){
19. return j;
20. }
21. int speedUp(int... z){
22. return k;
23. }
24. }
```

Exercises and Solutions – Execution Control Flow in Java

- Review Questions (Execution Control Flow in Java)
- Coding Questions

Review Questions :

1. Consider the following code fragment:

```
1 int i = 0;
2 do
3 {
4 System.out.println ( " I am in the do block.");
5 } while( i > 0)
```

What would be the output of this code fragment? (Choose all that apply.)

A. I am in the do block.
B. A compiler error occurs at line 5.
C. A runtime error occurs.
D. It compiles and runs but produces no output.

2. Consider the following code fragment:

```
1 for (int i = 0; i < 2; i++) {
2 for ( int j = 1; j < 4; j++) {
3 if ( i == j ) {
4 continue;
5 }
6 System.out.println ( " i = " + i + " j = " + j );
7 }
8 }
```

Which of the following lines would be part of the output? (Choose all that apply.)

A. i = 0 j = 1
B. i = 0 j = 2
C. i = 0 j = 3
D. i = 1 j = 1
E. i = 1 j = 2
F. i = 1 j = 3

3. Consider the following piece of code:

```
1 OuterLoop: for (int i = 0; i < 2; i++) {
2 for ( int j = 1; j < 4; j++) {
3 if ( i == j ) {
4 continue OuterLoop;
5 }
6 System.out.println ( " i = " + i + " j = " + j );
7 }
8 }
```

Which of the following lines would be part of the output? (Choose all that apply.)

A. i = 0 j = 1
B. i = 0 j = 2
C. i = 0 j = 3
D. i = 1 j = 1
E. i = 1 j = 2
F. i = 1 j = 3

4. Consider the following piece of code:

```
1 long i = 2;
2 switch (i) {
3 case 1:
4 System.out.println ("Case 1");
5 case 2:
6 System.out.println ("Case 2");
7 case 3:
8 System.out.println ("Case 3");
9 default:
10 System.out.println ("Default");
11 }
```

Which of the following lines will be included in the output? (Choose all that apply.)

A. Case 1
B. Case 2
C. Case 3
D. Default
E. There will be a compiler error due to line 1.

5. Consider the following code fragment:

```
1 int i = 0;
2 do
3 {
4 System.out.println ( " I am in the do block.");
5 } while( i > 0);
```

What would be the output from this code fragment?

A. I am in the do block.
B. A compiler error occurs at line 5.
C. A runtime error occurs.
D. It compiles and runs but produces no output.

6. Consider the following code fragment:

```
1 int i = 0;
2 while( i > 0)
3 {
4 System.out.println ( " I am in the do block.");
5 }
```

What would be the output from this code fragment?

A. I am in the do block.
B. A compiler error occurs at line 2.
C. A runtime error.
D. It compiles and runs but produces no output.

7. Consider the following piece of code:

```
1 int i = 1, j=1;
2 switch (i + j) {
3 case 1:
4 System.out.println ("Case 1");
5 case 2:
6 System.out.println ("Case 2");
7 case 3:
8 System.out.println ("Case 3");
9 default:
10 System.out.println ("Default");
11 }
```

Which of the following statements is true about this piece of code?

A. The code will not compile due to line 2.
B. The output would be Case 2.
C. The output would be Case 2 followed by Case 3.
D. The output would be Case 2 followed by Case 3 followed by Default.
E. There will be a compiler error due to line 1.

8. Consider the following code:

```
1. class ContinueTest {
2. public static void main(String[] arg)
3. {
4. int i = 2;
5. Outer:
6. if ( i < 5 ) {
7. System.out.println("I: " + i);
8. i++;
9. continue Outer;
10. }
11. }
12. }
```

What is the result?

A. A compiler error occurs at line 9.
B. I: 2
I: 3
I: 4
C. I: 2
D. The program compiles but throws an exception when executed.

9. Consider the following code fragment:

```
1. class RevQOne{
2. public static void main(String [] args) {
3. boolean i = true;
4. boolean j = false;
5. short k = 10;
6. if((k == 10) && (j = true))k--;
7. if((i = false) || ( k == 9))
8. k--;
9. k--;
10. System.out.println("k=" + k);
11. }
12. }
```

What is the result?

A. Compilation fails.
B. K=7
C. K=8
D. k=9
E. k=10
F. An exception is thrown at runtime.

10. Consider the following code:

```
1. class RevQTwo{
2. static int[] myArray = new int[3];
3. public static void main(String [] args) {
4. myArray[0]=1; myArray[1]=2; myArray[2]=3;
5. for(int i : myArray)
6. System.out.print(i);
7. }
8. }
```

Which of the following inserted in line 5 will generate the output 123?

A. for(int[] i : myArray)
B. for(int i : myArray)
C. for(myArray : int i)
D. for(int I : myArray.iterator())

11. Consider the code in Listing 6-4. What is the result?

A. i=5 j=3 x=true y=false
B. i=6 j=4 x=false y=true
C. i=5 j=3 x=false y=true
D. Compilation fails at lines 9 and 10.

12. Consider the following code:

```
1. for (int i = 0; i < 2; i++) {
2. for (int j = 0; j < 3; j++) {
3. if (i == j) {
4. continue;
5. }
6. System.out.println("i = " + i + " j = " + j);
7. }
8. }
```

Which lines would be part of the output? (Choose all that apply.)

A. i = 0 j = 0
B. i = 0 j = 1
C. i = 0 j = 2
D. i = 1 j = 0
E. i = 1 j = 1
F. i = 1 j = 2

13. Consider the following code:

```
1. outer: for (int i = 0; i < 2; i++) {
2. for (int j = 0; j < 3; j++) {
3. if (i == j) {
4. continue outer;
5. }
6. System.out.println("i = " + i + " j = " + j);
7. }
8. }
```

Which lines would be part of the output? (Choose all that apply.)

A. i = 0 j = 0
B. i = 0 j = 1
C. i = 0 j = 2
D. i = 1 j = 0
E. i = 1 j = 1
F. i = 1 j = 2

14. Which of the following are legal loop constructions? (Choose all that apply.)

A.
```
while (int i<7) {
i++;
System.out.println("i is " + i);
}
```

B. int i = 3;
while (i) {
System.out.println("i is " + i);
}

C. int j = 0;
for (int k=0, j+k != 10; j++,k++) {
System.out.println("j=" + j + ", k=" + k);
}

D. int j=0;
do {
System.out.println("j=" + j++);
if (j==3)
continue loop;
} while (j<10);

15. What would be the output from this code fragment?

1. int x = 0, y = 4, z = 5;
2. if (x > 2) {
3. if (y < 5) {
4. System.out.println("message one");
5. }
6. else {
7. System.out.println("message two");
8. }
9. }
10. else if (z > 5) {
11. System.out.println("message three");
12. }
13. else {
14. System.out.println("message four");
15. }

A. message one
B. message two
C. message three
D. message four

16. Which statement is true about the following code fragment?

```
1. int j = 2;
2. switch (j) {
3. case 2:
4. System.out.println("value is two");
5. case 2 + 1:
6. System.out.println("value is three");
7. break;
8. default:
9. System.out.println("value is " + j);
10. break;
11. }
```

A. The code is illegal because of the expression at line 5.
B. The acceptable types for the variable j, as the argument to the switch() construct, could be any of byte, short, int, or long.
C. The output would be the text value is two.
D. The output would be the text value is two followed by the text value is three.
E. The output would be the text value is two, followed by the text value is three, followed by the text value is 2.

17. Suppose salaries is an array containing floats. Which of the following are valid loop control statements for processing each element of salaries?

A. for (float f:salaries)
B. for (int i:salaries)
C. for (float f::salaries)
D. for (int i::salaries)

18. Which of the following are legal? (Choose all that apply.)

A. for (int i=0, j=1; i<10; i++, j++)
B. for (int i=0, j=1;; i++, j++)
C. for (int i=0, float j=1; ; i++, j++)
D. for (String s = “”; s.length()<10; s += ‘!’)

19. Which of the following are legal loop definitions? (Choose all that apply.)

A. while (int a = 0) { /* whatever */ }
B. while (int a == 0) { /* whatever */ }
C. do { /* whatever */ } while (int a = 0)
D. do { /* whatever */ } while (int a == 0)
E. for (int a==0; a<100; a++) { /* whatever */ }
F. None of them are legal.

20. Which of the following are legal argument types for a switch statement?

A. byte
B. int
C. long
D. float
E. char
F. String

Solutions:

1. Answer: B

A colon (;) is required at the end of the while statement.

2. Answer: A, B, C, E, and F

The output in D will never be printed due to the continue statement.

3. Answer: A, B, and C

When the values of i and j are equal, the execution control jumps back to line 1.

4. Answer: E

The argument of switch must be of type byte, short, char, or int. The compiler will give the error at line 2, which you can avoid by changing the type of the variable i in line 1.

5. Answer: A

The do block will be executed at least once even if the condition is false.

6. Answer: D

The condition in while() is false, so execution control will not enter the while block.

7. Answer: D

There is no break statement, so a fall through will occur.

8. Answer: A
The continue statement can only exist inside a loop.

9. Answer: A

In line 6, the first condition is true, so the second condition is checked, which turns out to be true as well because j is set equal to true. So, k is decremented by 1. In line 7, i is set to false, so the first
condition turns out to be false and the second condition is tested, which is true. Therefore, k is decremented again. Line 9 will always be executed because this is independent of the if statement.

10. Answer: B

B is the correct answer because it is a for-each loop.

11. Answer: C

In line 7, i is incremented after testing the condition, so y is set to true and i is incremented again. In line 8, j is incremented before testing the first condition. Because the first condition turns out to be false, the second condition is not even tested. In line 9, x is set to false, and because it is an OR, the second condition is evaluated.

12. Answer: B, C, D, F.

The loops iterate i from 0 to 1 and j from 0 to 2. However, the inner loop executes a continue statement whenever the values of i and j are the same. Because the output is generated inside the inner loop, after the continue statement, no output is generated when the values are the same. Therefore, the outputs suggested by options A and E are skipped.

13. Answer: D.

It seems that the variable i will take the values 0 and 1, and for each of these values, j will take values 0, 1, and 2. However, whenever i and j have the same value, the outer loop is continued before the output is generated. Because the outer loop is the target of the continue statement, the whole of the inner loop is abandoned. Therefore, the only line to be output is that shown in option D.

14. Answer: C.

In A, the variable declaration for i is illegal. This type of declaration is permitted only in the first part of a for() loop. In B, the loop control expression—the variable i in this case—is of type int. A boolean expression is required. C is valid. Despite the complexity of declaring one value inside the for() construction and one outside (along with the use of the comma operator in the end part), this code is entirely legitimate. D would be correct, except that the label has been omitted from the 2nd line, which should read loop: do {.

15. Answer: D.

The first test at line 2 fails, which immediately causes control to skip to line 10, bypassing both the possible tests that might result in the output of message one or message two. So, even though the test at line 3 would be true, it is never made; A is not correct. At line 10, the test is again false, so the message at line 11 is skipped, but message four, at line 14, is output.

16. Answer: D.

A is incorrect because the code is legal despite the expression at line 5; the expression itself is a constant. B is incorrect because it states that the switch() part can take a long argument. Only byte, short, char, and int are acceptable. The output results from the value 2 like this: first, the option case 2: is selected, which outputs value is two. However, there is no break statement between lines 4 and 5, so the execution falls into the next case and outputs value is three from line 6. The default: part of a switch() is executed only when no other options have been selected, or if no break precedes it. Neither of these situations holds true, so the output consists only of the two messages listed in D.

17. Answer: A.

Option A demonstrates the correct syntax of an enhanced for loop, traversing the elements of an array of floats.

18. Answer:

A, B, D. A and B demonstrate multiple initialization and increment parts, which are legal. In B, the test part is empty, so the loop will run forever unless it hits a break statement, throws an exception, or does something equally catastrophic. C is illegal because only one type may be declared in the initialization. D is unusual because it uses strings rather than the more commonly seen ints, but it is perfectly legal.

19. Answer:

F. A through D are all illegal because only for loops allow loop variables to be declared in the loop control code. E is illegal because the variable must be initialized with =, not ==.

20. Answer: A, B, E.

The argument of a switch statement must be a byte, char, short, int, or enum. Enums are discussed in Chapter 6, “Objects and Classes.”

Coding Questions:

- Compile and Run the following code:
- Switch the variables on line 6 around. What is the result? Why?
- Change b1 and b2 to ints and assign values to them. What happens when you try to compile?
- What do you need to do on line 6 to make this code compile if b1 and b2 are ints (hint: compare the values rather than using an assignment statement)

```
1. class IfTest {
2. public static void main(String[] args)
3. {
4. boolean b1 = false;
5. boolean b2 = true;
6. if(b1=b2){
7. System.out.println("The value of b1: " + b1);
8. }
9. }
10.}
```

- Compile, Run and Understand the following code:
- When running the code try the following commands
  - java SwitchTest yellow
  - java SwitchTest green
  - java SwitchTest red

```
1. class SwitchTest {
2. public static void main(String[] args)
3. {
4. Signal sig = Enum.valueOf(Signal.class, args[0].toUpperCase());
5. switch(sig){
6. case RED:
7. sig.redSays();
8. break;
9. case YELLOW:
10. sig.yellowSays();
11. case GREEN:
12. sig.greenSays();
13. }
14. }
15. }
16. enum Signal {RED, YELLOW, GREEN;
17. public void redSays(){
18. System.out.println("STOP");
19. }
20. public void yellowSays(){
21. System.out.println("STOP if it is safe to do so.");
22. System.out.println("Otherwise");
```

```
23. }
24. public void greenSays(){
25. System.out.println("Keep going.");
26. }
27. }
```

- Compile, Run and Understand the following code:
- Create a second int array of size 10, initialise its variabales as you like and use a for-each loop to iterate through it and print out its values.

```
1. class ForEachTest {
2. public static void main(String[] args) {
3. int[] myArray = new int[3];
4. myArray[0]= 10;
5. myArray[1] = 20;
6. myArray[2] = 30;
7. for(int i : myArray) {
8. System.out.println (i);
9. }
10. }
11. }
```

- Compile, Run and Understand the following code:

```
1. class CodeWalkFive {
2. public static void main(String [] args) {
3. boolean x = true;
4. boolean y = false;
5. int i = 1;
6. int j = 1;
7. if((i++ == 1) && (y = true))i++;
8. if((++j == 1) && (x = false))j++;
9. if((x = false) || (++i == 4))i++;
10. if((y = true) || (++j == 4))j++;
11. System.out.print("i=" + i);
12. System.out.print(" j=" + j);
13. System.out.print(" x=" + x);
14. System.out.print(" y=" + y);
15. }
16. }
```