The OCPJP 7 Exam: Pretest

Time: **1 hour 15 minutes** ***No. of questions:*** **45**

---

1. **Consider the following program:**

```
class StrEqual {
        public static void main(String []args) {
               String s1 = "hi";
               String s2 = new String("hi");
               String s3 = "hi";

               if(s1 == s2) {
                       System.out.println("s1 and s2 equal");
               } else {
                       System.out.println("s1 and s2 not equal");
               }

               if(s1 == s3) {
                       System.out.println("s1 and s3 equal");
               } else {
                       System.out.println("s1 and s3 not equal");
               }
        }
}
```

Which one of the following options provides the output of this program when executed?

a)
s1 and s2 equal
s1 and s3 equal
b)
s1 and s2 equal
s1 and s3 not equal
c)
s1 and s2 not equal
s1 and s3 equal
d)
s1 and s2 not equal
s1 and s3 not equal

---

2. **Consider the following program:**

```
class Point2D {
        private int x, y;
        public Point2D(int x, int y) {
                x = x;
        }

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

```
        public static void main(String []args) {
                Point2D point = new Point2D(10, 20);
                System.out.println(point);
        }
}
```

Which one of the following options provides the output of this program when executed?

a) point
b) Point
c) [0, 0]
d) [10, 0]
e) [10, 20]

---

3. **Consider the following program:**

```
class Increment {
        public static void main(String []args) {
               Integer i = 10;
               Integer j = 11;
               Integer k = ++i;         // INCR
               System.out.println("k == j is " + (k == j));
               System.out.println("k.equals(j) is " + k.equals(j));
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) When executed, this program prints
   k == j is false
   k.equals(j) is false
b) When executed, this program prints
   k == j is true
   k.equals(j) is false
c) When executed, this program prints
   k == j is false
   k.equals(j) is true
d) When executed, this program prints
   k == j is true
   k.equals(j) is true
e) When compiled, the program will result in a compiler error in the line marked with the comment INCR.

---

4. **Consider the following program:**

```
class ArrayCompare {
        public static void main(String []args) {
               int []arr1 = {1, 2, 3, 4, 5};
               int []arr2 = {1, 2, 3, 4, 5};
               System.out.println("arr1 == arr2 is " + (arr1 == arr2));
               System.out.println("arr1.equals(arr2) is " + arr1.equals(arr2));
```

```
                System.out.println("Arrays.equals(arr1, arr2) is " +
                                java.util.Arrays.equals(arr1, arr2));
        }
    }
```

Which one of the following options provides the output of this program when executed?

a) arr1 == arr2 is false
   arr1.equals(arr2) is false
   Arrays.equals(arr1, arr2) is true
b) arr1 == arr2 is true
   arr1.equals(arr2) is false
   Arrays.equals(arr1, arr2) is true
c) arr1 == arr2 is false
   arr1.equals(arr2) is true
   Arrays.equals(arr1, arr2) is true
d) arr1 == arr2 is true
   arr1.equals(arr2) is true
   Arrays.equals(arr1, arr2) is false
e) arr1 == arr2 is true
   arr1.equals(arr2) is true
   Arrays.equals(arr1, arr2) is true

---

5. **Consider the following program:**

```
class NullInstanceof {
        public static void main(String []args) {
                String str = null;
                if(str instanceof Object)    // NULLCHK
                        System.out.println("str is Object");
                else
                        System.out.println("str is not Object");
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) This program will result in a compiler error in line marked with comment NULLCHK.
b) This program will result in a NullPointerException in line marked with comment NULLCHK.
c) When executed, this program will print the following: str is Object.
d) When executed, this program will print the following: str is not Object.

---

6. **Consider the following program:**

```
interface Side { String getSide(); }

class Head implements Side {
        public String getSide() { return "Head "; }
}

class Tail implements Side {
```

```
        public String getSide() { return "Tail "; }
}

class Coin {
        public static void overload(Head side) { System.out.print(side.getSide()); }
        public static void overload(Tail side) { System.out.print(side.getSide()); }
        public static void overload(Side side) { System.out.print("Side "); }
        public static void overload(Object side) { System.out.print("Object "); }

        public static void main(String []args) {
               Side firstAttempt = new Head();
               Tail secondAttempt = new Tail();
               overload(firstAttempt);
               overload((Object)firstAttempt);
               overload(secondAttempt);
               overload((Side)secondAttempt);
        }
}
```

What is the output of this program when executed?

a) Head Head Tail Tail
b) Side Object Tail Side
c) Head Object Tail Side
d) Side Head Tail Side

---

7. **Consider the following program:**

```
class Overloaded {
        public static void foo(Integer i) { System.out.println("foo(Integer)"); }
        public static void foo(short i) { System.out.println("foo(short)"); }
        public static void foo(long i) { System.out.println("foo(long)"); }
        public static void foo(int ... i) { System.out.println("foo(int ...)"); }
        public static void main(String []args) {
                foo(10);
        }
}
```

Which one of the following options correctly describes the output of this program?

a) `foo(Integer)`
b) `foo(short)`
c) `foo(long)`
d) `foo(int ...)`

---

8. **Consider the following program:**

```
class Base {
        public static void foo(Base bObj) {
               System.out.println("In Base.foo()");
               bObj.bar();
        }
```

```
        public void bar() {
                System.out.println("In Base.bar()");
        }
}

class Derived extends Base {
        public static void foo(Base bObj) {
                System.out.println("In Derived.foo()");
                bObj.bar();
        }
        public void bar() {
                System.out.println("In Derived.bar()");
        }
}

class OverrideTest {
        public static void main(String []args) {
                Base bObj = new Derived();
                bObj.foo(bObj);
        }
}
```

What is the output of this program when executed?

a)
In Base.foo()
In Base.bar()
b)
In Base.foo()
In Derived.bar()
c)
In Derived.foo()
In Base.bar()
d)
In Derived.foo()
In Derived.bar()

---

9. Consider the following program:

```
class CannotFlyException extends Exception {}

interface Birdie {
        public abstract void fly() throws CannotFlyException;
}

interface Biped {
        public void walk();
}

abstract class NonFlyer {
        public void fly() { System.out.print("cannot fly ");  }         // LINE A
}
```

```
class Penguin extends NonFlyer implements Birdie, Biped {          // LINE B
        public void walk() { System.out.print("walk "); }
}

class PenguinTest {
        public static void main(String []args) {
               Penguin pingu = new Penguin();
               pingu.walk();
               pingu.fly();
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) Compiler error in line with comment LINE A because `fly()` does not declare to throw `CannotFlyException`.
b) Compiler error in line with comment LINE B because `fly()` is not defined and hence need to declare it abstract.
c) It crashes after throwing the exception `CannotFlyException`.
d) When executed, the program prints "walk cannot fly".

---

10. Consider the following program:

```
class TestSwitch {
        public static void main(String []args) {
               String [] cards = { "Club", "spade", " diamond ", "hearts" };
               for(String card : cards) {
                       switch(card) {
                       case "Club" : System.out.print(" club "); break;
                       case "Spade" :  System.out.print(" spade "); break;
                       case "diamond" :  System.out.print(" diamond "); break;
                       case "heart" :  System.out.print(" heart "); break;
                       default: System.out.print(" none ");
                       }
               }
        }
}
```

Which one of the following options shows the output of this program?

a) none none none none
b) club none none none
c) club spade none none
d) club spade diamond none
e) club spade diamond heart

---

11. Consider the following program:

```
class Outer {
        static class Inner {
                public final String text = "Inner";
        }
}
```

```
class InnerClassAccess {
        public static void main(String []args) {
                System.out.println(/*CODE HERE*/);
        }
}
```

Which one of the following expressions when replaced for the text in place of the comment /*CODE HERE*/ will print the output "Inner" in console?

a) `new Outer.Inner().text`
b) `Outer.new Inner().text`
c) `Outer.Inner.text`
d) `new Outer().Inner.text`

---

12. **Consider the following enumeration definition:**

```
enum Cards { CLUB, SPADE, DIAMOND, HEARTS };

class CardsEnumTest {
        public static void main(String []args) {
               /* TRAVERSE */
        }
}
```

Which one of the following will you replace in place of the comment /* TRAVERSE */ **to traverse the** `Cards` **enumeration and print the output "CLUB SPADE DIAMOND HEARTS"?**

a)
```
for(Cards card : Cards.values())
      System.out.print(card + " ");
```
b)
```
for(Cards card : Cards.iterator())
      System.out.print(card + " ");
```
c)
```
for(Cards card : Cards.enums())
      System.out.print(card + " ");
```
d)
```
for(Cards card : Cards.items())
      System.out.print(card + " ");
```
e) There is no way to print the string names of this enumeration. The `toString()` method of enumeration returns the ordinal value of the enumeration, which is equivalent to calling `card.ordinal().toString();`.

---

13. **Given these three definitions**

```
interface I1 {}
interface I2 {}
abstract class C {}
```

which one of the following will compile without errors?

a) `class CI12 extends C, I1, I2 {}`
b) `class CI12 implements C extends I1, I2 {}`
c) `class CI12 implements C, I1, I2 {}`
d) `class CI12 extends C implements I1, I2 {}`
e) `class CI12 extends C implements I1 implements I2 {}`
f) `class CI12 implements C extends I1 extends I2 {}`

14. Given these two definitions

```
interface I1 {}
interface I2 {}
```

which one of the following will compile without errors?

a) `interface II implements I1, I2 {}`
b) `interface II implements I1 implements I2 {}`
c) `interface II implements I1 extends I2 {}`
d) `interface II extends I1, I2 {}`

---

15. Consider the following program:

```
abstract class AbstractBook {
        public String name;
}

interface Sleepy {
        public String name = "undefined";
}

class Book extends AbstractBook implements Sleepy {
        public Book(String name) {
                this.name = name;             // LINE A
        }
        public static void main(String []args) {
                AbstractBook philosophyBook = new Book("Principia Mathematica");
                System.out.println("The name of the book is " + philosophyBook.name); // LINE B
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) The program will print the output "The name of the book is Principia Mathematica".
b) The program will print the output "The name of the book is undefined".
c) The program will not compile and result in a compiler error "ambiguous reference to name" in line marked with comment LINE A.
d) The program will not compile and result in a compiler error "ambiguous reference to name" in line marked with comment LINE B.

---

16. Which one of the following relationships describes the OO design concept of "composition"?

a) is-a
b) is-a-kind-of
c) has-a
d) is-implemented-in-terms-of
e) composed-as
f) DAO

17. **Consider the following program:**

```
import java.util.Arrays;

class DefaultSorter {
        public static void main(String[] args) {
               String[] brics = {"Brazil", "Russia", "India", "China"};
               Arrays.sort(brics, null);       // LINE A
               for(String country : brics) {
                       System.out.print(country + " ");
               }
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) This program will result in a compiler error in line marked with comment LINE A.
b) When executed, the program prints the following: Brazil Russia India China.
c) When executed, the program prints the following: Brazil China India Russia.
d) When executed, the program prints the following: Russia India China Brazil.
e) When executed, the program throws a runtime exception of `NullPointerException` when executing the line marked with comment LINE A.
f) When executed, the program throws a runtime exception of `InvalidComparatorException` when executing the line marked with comment LINE A.

---

18. **Consider the following program:**

```
import java.util.*;

class DequeTest {
        public static void main(String []args) {
               Deque<Integer> deque = new ArrayDeque<>();
               deque.addAll(Arrays.asList(1, 2, 3, 4, 5));
               System.out.println("The removed element is: " + deque.remove()); // ERROR?
        }
}
```

Which one of the following correctly describes the behavior of this program?

a) When executed, this program prints the following: "The removed element is: 5".
b) When executed, this program prints the following: "The removed element is: 1".
c) When compiled, the program results in a compiler error of "remove() returns void" for the line marked with the comment ERROR.
d) When executed, this program throws `InvalidOperationException.`

---

19. **Consider the following program:**

```
import java.util.*;

class Diamond {
        public static void main(String[] args) {
               List list1 = new ArrayList<>(Arrays.asList(1, "two", 3.0));  // ONE
```

```
                List list2 = new LinkedList<>
                        (Arrays.asList(new Integer(1), new Float(2.0F), new Double(3.0))); // TWO
                list1 = list2;  // THREE
                for(Object element : list1) {
                        System.out.print(element + " ");
                }
        }
}
```

Which one of the following describes the expected behavior of this program?

a) The program results in compiler error in line marked with comment `ONE`.
b) The program results in compiler error in line marked with comment `TWO`.
c) The program results in compiler error in line marked with comment `THREE`.
d) When executed, the program prints 1 2.0 3.0.
e) When executed, this program throws a `ClassCastException`.

---

20. Consider the following program:

```
class SimpleCounter<T> {
        private static int count = 0;
        public SimpleCounter() {
                count++;
        }
        static int getCount() {
                return count;
        }
}

class CounterTest {
        public static void main(String []args) {
                SimpleCounter<Double> doubleCounter = new SimpleCounter<Double>();
                SimpleCounter<Integer> intCounter = null;
                SimpleCounter rawCounter = new SimpleCounter();      // RAW
                System.out.println("SimpleCounter<Double> counter is "
                        + doubleCounter.getCount());
                System.out.println("SimpleCounter<Integer> counter is " + intCounter.getCount());
                System.out.println("SimpleCounter counter is " + rawCounter.getCount());
        }
}
```

Which one of the following describes the expected behavior of this program?

a) This program will result in a compiler error in the line marked with comment RAW.
b) When executed, this program will print
   SimpleCounter<Double> counter is 1
   SimpleCounter<Integer> counter is 0
   SimpleCounter counter is 1
c) When executed, this program will print
   SimpleCounter<Double> counter is 1
   SimpleCounter<Integer> counter is 1
   SimpleCounter counter is 1

d) When executed, this program will print
SimpleCounter<Double> counter is 2
SimpleCounter<Integer> counter is 0
SimpleCounter counter is 2
e) When executed, this program will print
SimpleCounter<Double> counter is 2
SimpleCounter<Integer> counter is 2
SimpleCounter counter is 2

---

21. Consider the following program:

```
class UsePrintf{
        public static void main(String []args) {
               int c = 'a';
               float f = 10;
               long ell = 100L;
              System.out.printf("char val is %c, float val is %f, long int val is %ld \n", c, f, ell);
        }
}
```

Which one of the following options best describes the behavior of this program when executed?

a) The program prints the following: char val is a, float val is 10.000000, long int val is 100.
b) The program prints the following: char val is 65, float val is 10.000000, long int val is 100.
c) The program prints the following: char val is a, float val is 10, long int val is 100L.
d) The program prints the following: char val is 65, float val is 10.000000, long int val is 100L.
e) The program prints the following: char val is 65, float val is 10, long int val is 100L.
f) The program throws an exception of `java.util.UnknownFormatConversionException: Conversion = 'l'`.

---

22. Consider the following program:

```
import java.util.regex.Pattern;

class Split {
        public static void main(String []args) {
               String date = "10-01-2012"; // 10th January 2012 in dd-mm-yyyy format
               String [] dateParts = date.split("-");
               System.out.print("Using String.split method: ");
               for(String part : dateParts) {
                       System.out.print(part + " ");
               }
               System.out.print("\nUsing regex pattern: ");
               Pattern datePattern = Pattern.compile("-");
               dateParts = datePattern.split(date);
               for(String part : dateParts) {
                       System.out.print(part + " ");
               }
        }
}
```

Which one of the following options correctly provides the output of this program?

a)
Using String.split method: 10-01-2012
Using regex pattern: 10 01 2012
b)
Using String.split method: 10 01 2012
Using regex pattern: 10 01 2012
c)
Using String.split method: 10-01-2012
Using regex pattern: 10-01-2012
d)
Using String.split method:
Using regex pattern: 10 01 2012
e)
Using String.split method: 10 01 2012
Using regex pattern:
f)
Using String.split method:
Using regex pattern:

---

23. Consider the following program:

```
import java.util.regex.Pattern;

class Regex {
        public static void main(String []args) {
               String pattern = "a*b+c{3}";
               String []strings = { "abc", "abbccc", "aabbcc", "aaabbbccc" };
               for(String str : strings) {
                       System.out.print(Pattern.matches(pattern, str) + " ");
               }
        }
}
```

Which one of the following options correctly shows the output of this program?

a) true true true true
b) true false false false
c) true false true false
d) false true false true
e) false false false true
f) false false false false

---

24. Consider the following program:

```
class MatchCheck {
        public static void main(String []args) {
               String[]strings = {"Severity 1", "severity 2", "severity3",
"severity five"};
               for(String str : strings) {
                       if(!str.matches("^severity[\\s+][1-5]")) {
```

```
                                System.out.println(str + " does not match");
                        }
                }
        }
}
```

Which one of the following options correctly shows the output of this program?

a)
Severity 1 does not match
severity 2 does not match
severity five does not match
b)
severity3 does not match
severity five does not match
c)
Severity 1 does not match
severity 2 does not match
d)
Severity 1 does not match
severity3 does not match
severity five does not match

---

25. Consider the following program:

```
import java.lang.*;

class InvalidValueException extends IllegalArgumentException {}
class InvalidKeyException extends IllegalArgumentException {}

class BaseClass {
        void foo() throws IllegalArgumentException {
                throw new IllegalArgumentException();
        }
}

class DeriClass extends BaseClass {
        public void foo() throws IllegalArgumentException {
                throw new InvalidValueException();
        }
}

class DeriDeriClass extends DeriClass {
        public void foo() {           // LINE A
                throw new InvalidKeyException();
        }
}
```

```
class EHTest {
        public static void main(String []args) {
               try {
                       BaseClass base = new DeriDeriClass();
                       base.foo();
               } catch(RuntimeException e) {
                       System.out.println(e);
               }
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) The program prints the following: `InvalidKeyException`.
b) The program prints the following: `RuntimeException`.
c) The program prints the following: `IllegalArgumentException`.
d) The program prints the following: `InvalidValueException`.
e) When compiled, the program will result in a compiler error in line marked with comment Line A due to missing throws clause.

---

26. Consider the following program:

```
class EHBehavior {
        public static void main(String []args) {
               try {
                        int i = 10/0; // LINE A
                        System.out.print("after throw -> ");
                } catch(ArithmeticException ae) {
                        System.out.print("in catch -> ");
                        return;
                } finally {
                        System.out.print("in finally -> ");
                }
                System.out.print("after everything");
        }
}
```

Which one of the following options best describes the behavior of this program?

a) The program prints the following: in catch -> in finally -> after everything.
b) The program prints the following: after throw -> in catch -> in finally -> after everything.
c) The program prints the following: in catch -> in finally -> after everything.
d) The program prints the following: in catch -> after everything.
e) The program prints the following: in catch -> in finally ->.
f) When compiled, the program results in a compiler error in line marked with comment in LINE A for divide-by-zero.

27. **Consider the following program:**

```
class AssertionFailure {
        public static void main(String []args) {
               try {
                       assert false;
               } catch(RuntimeException re) {
                       System.out.println("RuntimeException");
               } catch(Exception e) {
                       System.out.println("Exception");
               } catch(Error e) {   // LINE A
                       System.out.println("Error" + e);
               } catch(Throwable t) {
                       System.out.println("Throwable");
               }
        }
}
```

This program is invoked in command line as follows:

```
java AssertionFailure
```

Choose one of the following options describes the behavior of this program:

a) Compiler error at line marked with comment LINE A
b) Prints “RuntimeException” in console
c) Prints “Exception”
d) Prints “Error”
e) Prints “Throwable”
f) Does not print any output on console

---

28. **Consider the following program:**

```
import java.io.*;

class CreateFilesInFolder {
        public static void main(String []args) {
               String[] fileList = { "/file1.txt", "/subdir/file2.txt", "/file3.txt" };
               for (String file : fileList) {
                       try {
                              new File(file).mkdirs();
                       }
                       catch (Exception e) {
                              System.out.println("file creation failed");
                              System.exit(-1);
                       }
               }
        }
}
```

Assume that underlying file system has necessary permissions to create files, and that the program executed successfully without printing the message "file creation failed." (In the answers, note that the term "current directory" means the directory from which you execute this program, and the term "root directory" in Windows OS means the root path of the current drive from which you execute this program.) What is the most likely behavior when you execute this program?

a) This program will create file1.txt and file3.txt files in the current directory, and file2.txt file in the subdir directory of the current directory.
b) This program will create file1.txt and file3.txt directories in the current directory and the file2.txt directory in the "subdir" directory in the current directory.
c) This program will create file1.txt and file3.txt files in the root directory, and a file2.txt file in the "subdir" directory in the root directory.
d) This program will create file1.txt and file3.txt directories in the root directory, and a file2.txt directory in the "subdir" directory in the root directory.

---

29. Which of the following two statements is true regarding object serialization in Java?

a) A `serializable` interface declares two methods, `readObject()` and `writeObject()`. To support serialization in your class, you need to implement the `Serializable` interface and define these two methods.
b) When serializing an object that has references to other objects, the serialization mechanism also includes the referenced objects as part of the serialized bytes.
c) When an object is serialized, the class members that are declared as `transient` will not be serialized (and hence their values are lost after deserialization).
d) The `Externalizable` interface is a marker interface; in other words, it's an empty interface that does not declare any methods.
e) If you attempt to serialize or persist an object that does not implement the `Externalizable` interface, you'll get a `NotExternalizableException`.

---

30. Consider the following program:

```
import java.util.*;

class Separate {
        public static void main(String []args) {
               String text = "<head>first program </head> <body>hello world</body>";
               Set<String> words = new TreeSet<>();
               try ( Scanner tokenizingScanner = new Scanner(text) ) {
                       tokenizingScanner.useDelimiter("\\W");
                       while(tokenizingScanner.hasNext()) {
                              String word = tokenizingScanner.next();
                              if(!word.trim().equals("")) {
                                     words.add(word);
                              }
                       }
               }
               for(String word : words) {
                       System.out.print(word + " ");
               }
        }
}
```

Which one of the following options correctly provides the output of this program?

a) hello body program head first world
b) body first head hello program world
c) head first program head body hello world body
d) head first program body hello world
e) < </ >

---

31. **Consider the following code snippet:**

```
Path wordpadPath = Paths.get("C:\\Program Files\\Windows NT\\Accessories\\wordpad.exe");
System.out.println(wordpadPath.subpath(beginIndex, endIndex));
```

What are the values of the integer values `beginIndex` and `endIndex` in this program that will result in this code segment printing the string "Program Files" as output?

a) beginIndex = 1 and endIndex = 2
b) beginIndex = 0 and endIndex = 1
c) beginIndex = 1 and endIndex = 1
d) beginIndex = 4 and endIndex = 16

---

32. **Consider the following program:**

```
import java.io.IOException;
import java.nio.file.*;

class Matcher {
        public static void main(String []args) {
               Path currPath = Paths.get(".");
               try (DirectoryStream<Path> stream =
                        Files.newDirectoryStream(currPath, "*o*?{java,class}")) {
                        for(Path file : stream) {
                                System.out.print(file.getFileName() + " ");
                        }
               } catch (IOException ioe) {
                        System.err.println("An I/O error occurred... exiting ");
               }
        }
}
```

Assume that the current path in which the program is run has the following files: `Copy.class`, `Copy.java`, `Dir.class`, `Dir.java`, `Hello.class`, `hello.html`, `Matcher.class`, `Matcher.java`, `OddEven.class`, and `PhotoCopy.java`. **Assuming that the program ran without throwing** `IOException`. **Which one of the following options correctly describes the behavior of this program when it is executed?**

a) Prints the following: `Copy.class Copy.java Hello.class hello.html OddEven.class PhotoCopy.java`
b) Prints the following: `Copy.class Copy.java PhotoCopy.java`
c) Prints the following: `Hello.class hello.html OddEven.class PhotoCopy.java`
d) Prints the following: `Copy.class Copy.java Hello.class OddEven.class PhotoCopy.java`
e) Prints the following: `PhotoCopy.java`
f) Does not print any output in console
g) Throws the exception `java.util.regex.PatternSyntaxException` because the pattern is invalid.

33. **Which one of the following options is a correct way to create a watch service for watching a directory for changes?**

a) `Watchable watch = FileSystems.getDefault().newWatchable();`
b) `WatchService watcher = FileSystems.getDefault().newWatchService();`
c) `DirectoryWatchService dirWatcher = FileSystems.getDefault().newDirectoryWatchService();`
d) `FileWatchService fileWatcher = FileSystems.getNewFileWatchService();`
e) `FileDirectoryWatchService fileDirWatcher = WatchService.getNewFileDirectoryWatchService();`

34. **Which of the following two statements are true regarding Statement and its derived types?**

a) Objects of type `Statement` can handle IN, OUT, and INOUT parameters.
b) `PreparedStatement` is used for executing stored procedures.
c) You can get an instance of `PreparedStatement` by calling `preparedStatement()` method in the `Connection` interface.
d) `CallableStatement` extends the `PreparedStatement` class; `PreparedStatement` in turn extends the `Statement` class.
e) The interface `Statement` and its derived interfaces implement the `AutoCloseable` interface, hence it can be used with try-with-resources statement.

35. **Consider the following sequence of statements when using JDBC API. Assume that you've a TempSensor table with the column name temp.**

```
// assume that connection is successfully established to the database
connection.setAutoCommit(true);
Statement statement = connection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
resultSet = statement.executeQuery("SELECT * FROM TempSensor");

// assume that the initial value of temp is "0" in the table

resultSet.moveToInsertRow();
resultSet.updateString("temp", "100");
resultSet.insertRow();
Savepoint firstSavepoint = connection.setSavepoint();

resultSet.moveToInsertRow();
resultSet.updateString("temp", "200");
resultSet.insertRow();
Savepoint secondSavepoint = connection.setSavepoint();

resultSet.moveToInsertRow();
resultSet.updateString("temp", "300");
resultSet.insertRow();
Savepoint thirdSavepoint = connection.setSavepoint();

connection.rollback(secondSavepoint);
connection.commit();
```

Which one of the following options correctly describes the behavior of this program?

a) temp value will be set to "100" in the table `TempSensor`.
b) temp value will be set to "200" in the table `TempSensor`.
c) temp value will be set to "300" in the table `TempSensor`.
d) temp value will be set to "0" in the table `TempSensor`.
e) The program will result in throwing a `SQLException` because auto-commit is true.

---

36. Which one of the following options correctly creates a JdbcRowSet object?

a)
```
RowSetProvider rowSetProvider = RowSetFactory.newProvider();
JdbcRowSet rowSet = rowSetProvider.createJdbcRowSet();
```
b)
```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();
JdbcRowSet rowSet = rowSetFactory.createJdbcRowSet();
```
c)
```
JdbcRowSet rowSet = RowSetProvider.newFactory().getJdbcRowSetInstance();
```
d)
```
JdbcRowSet rowSet = RowSetFactory.newProvider().getInstance(connection, "JdbcRowSet");
```

---

37. Consider the following program:

```
class Worker extends Thread {
        public void run()  {
               System.out.println(Thread.currentThread().getName());
        }
}

class Master {
        public static void main(String []args) throws InterruptedException {
               Thread.currentThread().setName("Master ");
               Thread worker = new Worker();
               worker.setName("Worker ");
               worker.start();
               Thread.currentThread().join();
               System.out.println(Thread.currentThread().getName());
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) When executed, the program prints the following: "Worker Master ".
b) When executed, the program prints "Worker ", and after that the program hangs (i.e., does not terminate).
c) When executed, the program prints "Worker " and then terminates.
d) When executed, the program throws `IllegalMonitorStateException`.
e) The program does not compile and fails with multiple compiler errors.

---

38. Which of the following two statements are true regarding the sleep() method defined in Thread class?

a) The `sleep()` method takes milliseconds as an argument for the time to sleep.
b) The `sleep()` method takes microseconds as an argument for the time to sleep.
c) The `sleep()` method relinquishes the lock when the thread goes to sleep and reacquires the lock when the thread wakes up.
d) The `sleep()` method can throw `InterruptedException` if it is interrupted by another thread when it sleeps.

39. **Consider the following program:**

```
class Waiter extends Thread {
        public static void main(String[] args) {
               new Waiter().start();
        }
        public void run() {
               try {
                       System.out.println("Starting to wait");
                       wait(1000);
                       System.out.println("Done waiting, returning back");
               }
               catch(InterruptedException e) {
                       System.out.println("Caught InterruptedException ");
               }
               catch(Exception e) {
                       System.out.println("Caught Exception ");
               }
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) The program prints
   Starting to wait
   Done waiting, returning back
b) The program prints
   Starting to wait
   Caught InterruptedException
c) The program prints
   Starting to wait
   Caught Exception
d) The program prints
   Starting to wait
   After that, the program gets into an infinite wait and deadlocks

40. **Consider the following program:**

```
import java.util.*;
import java.util.concurrent.*;

class SetTest {
        public static void main(String []args) {
               List list = Arrays.asList(10, 5, 10, 20);   // LINE A
               System.out.println(list);
               System.out.println(new HashSet(list));
               System.out.println(new TreeSet(list));
               System.out.println(new ConcurrentSkipListSet(list));
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) The program prints
[10, 5, 10, 20]
[20, 5, 10]
[5, 10, 20]
[5, 10, 20]
b) The program prints
[10, 5, 10, 20]
[5, 10, 20]
[5, 10, 20]
[20, 5, 10]
c) The program prints
[5, 10, 20]
[5, 10, 20]
[5, 10, 20]
[5, 10, 20]
d) The program prints
[10, 5, 10, 20]
[20, 5, 10]
[5, 10, 20]
[20, 5, 10]
e) The program prints
[10, 5, 10, 20]
[5, 10, 10, 20]
[5, 10, 20]
[10, 5, 10, 20]
f) Compiler error in line marked by the comment LINE A since `List` is not parameterized with the type `<Integer>`.

---

41. **Consider the following program:**

```
import java.util.concurrent.*;
import java.util.*;

class COWArrayListTest {
        public static void main(String []args) {
               ArrayList<Integer> aList = new CopyOnWriteArrayList<Integer>(); // LINE A
               aList.addAll(Arrays.asList(10, 20, 30, 40));
               System.out.println(aList);
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) When executed the program prints the following: [10, 20, 30, 40].
b) When executed the program prints the following: CopyOnWriteArrayList.class.
c) The program does not compile and results in a compiler error in line marked with comment `LINE A`.
d) When executed the program throws a runtime exception `ConcurrentModificationException`.
e) When executed the program throws a runtime exception `InvalidOperationException`.

42. **Consider the following program:**

```
import java.util.concurrent.*;
import java.util.*;

class Blocking {
        Deque<String> gangOfFour = new LinkedBlockingDeque<String>();
        class Producer extends Thread {
                String []authors = { "E Gamma", "R Johnson", "R Helm", "J Vlissides" };
                public void run() {
                        for(String author : authors) {
                                gangOfFour.add(author);
                                try {
                                        // take time to add
                                        Thread.sleep(1000);
                                }
                                catch(InterruptedException ie) {
                                        // ignore it
                                }
                        }
                }
        }

        class Consumer extends Thread {
                int numOfAuthors = 4;
                int processedAuthors = 0;
                public void run() {
                        while(processedAuthors < 4) {
                                while (gangOfFour.isEmpty()) { /*wait till an entry is inserted*/ }

                                System.out.println(gangOfFour.remove());
                                processedAuthors++;
                        }
                }
        }

        public static void main(String []args) {
                Blocking blocking = new Blocking();
                blocking.new Producer().start();
                blocking.new Consumer().start();
        }
}
```

Which one of the following options correctly describes the behavior of this program?

a) Prints
E Gamma
and then the program terminates.

b) Prints
E Gamma
R Johnson
R Helm
J Vlissides
and then the program enters a deadlock and never terminates.

c) Prints
E Gamma
R Johnson
R Helm
J Vlissides
and then the program terminates.
d) Prints
J Vlissides
R Helm
R Johnson
E Gamma
and then the program terminates.
e) The program does not print any output, enters a deadlock, and never terminates.

---

43. For localization, resource bundle property files are created that consist of key-value pairs. Which one of the following is a valid key value pair as provided in a resource bundle property file for some strings mapped to German language?

a)
```
<pair> <key>from</key> <value>von</value> </pair>
<pair> <key>subject</key> <value> betreff </value> </pair>
```
b)
```
from=von
subject=betreff
```
c)
```
key=from; value=von
key=subject; value=betreff
```
d)
```
pair<from,von>
pair<subject,betreff>
```

---

44. Assume that you've the following resource bundles in your classpath:

```
ResourceBundle.properties
ResourceBundle_ar.properties
ResourceBundle_en.properties
ResourceBundle_it.properties
ResourceBundle_it_IT_Rome.properties
```

Also assume that the default locale is English (US), where the language code is en and country code is US. Which one of these five bundles will be loaded for the call
`loadResourceBundle("ResourceBundle", new Locale("fr", "CA", ""));`**?**

a) `ResourceBundle.properties`
b) `ResourceBundle_ar.properties`
c) `ResourceBundle_en.properties`
d) `ResourceBundle_it.properties`
e) `ResourceBundle_it_IT_Rome.properties`

---

45. **Which one of the following is the correct implementation of a custom time formatter implementation that prints the current time in the format 10:42:30 where 10 is hours (value in range 1–12), 42 is minutes, and 30 is seconds?**

a) `System.out.println(new SimpleDateFormat("hh:mm:ss").format(new Date()));`
b) `System.out.println(new CustomDateFormat("hh:mm:ss").format(new Date()));`
c) `System.out.println(new SimpleTimeFormat("hh:mm:ss").format(new Date()));`
d) `System.out.println(new CustomDateTimeFormat("HH:MM:SS").format(new Date()));`

Answer sheet

Q.No	Answer	Q.No	Answer
1		24	
2		25	
3		26	
4		27	
5		28	
6		29	
7		30	
8		31	
9		32	
10		33	
11		34	
12		35	
13		36	
14		37	
15		38	
16		39	
17		40	
18		41	
19		42	
20		43	
21		44	
22		45	
23			

Answers with Explanations

1. c)

   s1 and s2 not equal
   s1 and s3 equal

   JVM sets a constant pool in which it stores all the string constants used in the type. If two references are declared with a constant, then both refer to the same constant object. The == operator checks the similarity of objects itself (and not the values in it). Here, the first comparison is between two distinct objects, so we get `s1 and s2 not equal`. On the other hand, since references of `s1` and `s3` refer to the same object, we get `s1 and s3 equal`.

2. c) [0, 0]

   The assignment `x = x;` inside the construct reassigns the passed parameter; it does *not* assign the member `x` in `Point2D`. The correct way to perform the assignment is `this.x = x;`. Field `y` is not assigned, so its value remains 0.

3. d) When executed, this program prints

   k == j is true
   k.equals(j) is true

   The `Integer` objects are immutable objects. If there is an `Integer` object for a value that already exists, then it does not create a new object again. In other words, Java uses sharing of immutable `Integer` objects, so two `Integer` objects are equal if their values are equal (no matter if you use == operators to compare the references or use `equals()` method to compare the contents).

4. a) arr1 == arr2 is false

   arr1.equals(arr2) is false
   Arrays.equals(arr1, arr2) is true

   The first comparison between two array objects is carried out using the == operator, which compares object similarity so it returns false here. The `equals()` method, which compares this array object with the passed array object, does not compare values of the array since it is inherited from the `Object` class. Thus we get another false. On the other hand, the `Arrays` class implements various `equals()` methods to compare two array objects of different types; hence we get true from the last invocation.

5. d) When executed, this program will print the following: str is not Object.

   The variable `str` was declared but not instantiated; hence the instanceof operator returns false.

6. b) Side Object Tail Side

   Overloading is based on the static type of the objects (while overriding and runtime resolution resolves to the dynamic type of the objects). Here is how the calls to the `overload()` method are resolved:

   - `overload(firstAttempt);` --> `firstAttempt` is of type `Side`, hence it resolves to `overload(Side)`.
   - `overload((Object)firstAttempt);` -> `firstAttempt` is casted to `Object`, hence it resolves to `overload(Object)`.

- `overload(secondAttempt);` -> `secondAttempt` is of type `Tail`, hence it resolves to `overload(Tail)`.
- `overload((Side)secondAttempt);` -> `secondAttempt` is casted to `Side`, hence it resolves to `overload(Side)`.

7. c) `foo(long)`

   For an integer literal, the JVM matches in the following order: `int`, `long, Integer, int...`. In other words, it first looks for an `int` type parameter; if it is not provided, then it looks for `long` type; and so on. Here, since the int type parameter is not specified with any overloaded method, it matches with `foo(long)`.

8. b)

```
In Base.foo()
In Derived.bar()
```

   A static method is resolved statically. Inside the static method, a virtual method is invoked, which is resolved dynamically.

9. d) When executed, the program prints "walk cannot fly".

   In order to override a method, it is not necessary for the overridden method to specify an exception. However, if the exception is specified, then the specified exception must be the same or a subclass of the specified exception in the method defined in the super class (or interface).

10. b) club none none none

    Here is the description of matches for the four enumeration values:

    - "club" matches with the case "Club".
    - For "Spade", the case "spade" does not match because of the case difference (switch case match is case sensitive).
    - does not match with "diamond" because case statements should exactly match and there are extra whitespaces in the original string.
    - "hearts" does not match the string "heart".

11. a) `new Outer.Inner().text`

    The correct way to access fields of the static inner class is to use the inner class instance along with the outer class, so `new Outer.Inner().text` will do the job.

12. a)
```
for(Cards card : Cards.values())
        System.out.print(card + " ");
```

    The `values()` method of an enumeration returns the array of enumeration members.

13. d) `class CI12 extends C implements I1, I2 {}`

    A class inherits another class using the `extends` keyword and inherits interfaces using the `implements` keyword.

14. d) `interface II extends I1, I2 {}`

    It is possible for an interface to extend one or more interfaces. In that case, we need to use the `extends` keyword and separate the list of super-interfaces using commas.

15. c) The program will not compile and will result in a compiler error "ambiguous reference to name" in LINE A.

    Since `name` is defined in both the base interface and the `abstract` class, any reference to the member `name` is ambiguous. The first reference to `name` is in the line marked with comment `LINE A`, so the error is flagged in this line by the compiler.

16. c) has-a

    Composition is a design concept that refers to the has-a relationship.

17. c) When executed, the program prints the following: Brazil China India Russia.

    When `null` is passed as a second argument to the `Arrays.sort()` method, it means that the default `Comparable` (i.e., natural ordering for the elements) should be used. The default `Comparator` results in sorting the elements in ascending order. The program does not result in a `NullPointerException` or any other exceptions or a compiler error.

18. b) When executed, this program prints the following: "The removed element is: 1".

    The `remove()` method is equivalent to the `removeFirst()` method, which removes the first element (head of the queue) of the `Deque` object.

19. d) When executed, the program prints the following: 1 2.0 3.0.

    The List is a generic type that is used here in raw form; hence it allows us to put different types of values in `list2`. Therefore, it prints the following: 1 2.0 3.0.

20. e) When executed, this program will print

    ```
    SimpleCounter<Double> counter is 2
    SimpleCounter<Integer> counter is 2
    SimpleCounter counter is 2
    ```

    `Count` is a static variable, so it belongs to the `class` and not to an instance. Each time constructor is invoked, `count` is incremented. Since two instances are created, the count value is two.

21. f) The program throws an exception for `java.util.UnknownFormatConversionException: Conversion = 'l'`

    There is no format specifier for long int, and the same `%d` format specifier for int is used for long as well. So, the format specifier `%ld` results in a runtime exception `UnknownFormatConversionException`.

22. b)

    ```
    Using String.split method: 10 01 2012
    Using regex pattern: 10 01 2012
    ```

    Using `str.split(regex)` is equivalent to using `Pattern.compile(regex).split(str)`.

23. d) false true false true

    Here are the following regular expression matches for the character x:

    - `x*` means matches with `x` for zero or more times.
    - `x+` means matches with `x` for one or more times.
    - `x{n}` means match `x` exactly `n` times.

The pattern `a*b+c{3}` means match `a` zero or more times, followed by `b` one or more times, and `c` exactly three times.

So, here is the match for elements in the `strings` array:

- For `"abc"`, the match fails, resulting in `false`.
- For `"abbccc"`, the match succeeds, resulting in `true`.
- For `"aabbcc"`, the match fails, resulting in `false`.
- For `"aaabbbccc"`, the match succeeds, resulting in `true`.

24. d) Severity 1 does not match.
severity3 does not match.
severity five does not match.

Here is the meaning of the patterns used:

`[^xyz]`	Any character except x, y, or z (i.e., negation)
`\s`	A whitespace character
`[a-z]`	from a to z

So the pattern `"^severity[\\s+][1-5]"` matches the string "severity" followed by whitespace followed by one of the letters 1 to 5.

For this pattern,

- "Severity 1" does not match because of the capital S in "Severity".
- "severity 2" matches.
- "severity3" does not match since there is no whitespace between severity and 3.
- "severity five" does not match since "five" does not match a numeral from 1 to 5.

25. a) The program prints the following: InvalidKeyException.

It is not necessary to provide an `Exception` thrown by a method when the method is overriding a method defined with an exception (using the throws clause). Hence, the given program will compile successfully and it will print `InvalidKeyException`.

26. e) The program prints the following: in catch -> in finally ->.

The statement `println("after throw -> ");` will never be executed since the line marked with the comment `LINE A` throws an exception. The catch handles `ArithmeticException`, so `println("in catch -> ");` will be executed. Following that, there is a return statement, so the function returns. But before the function returns, the `finally` statement should be called, hence the statement `println("in finally -> ");` will get executed. So, the statement `println("after everything");` will never get executed.

27. f) Does not print any output on the console

By default, assertions are disabled. If `-ea` (or the `-enableassertions` option to enable assertions), then the program would have printed "Error" since the exception thrown in the case of assertion failure is `java.lang.AssertionError`, which is derived from the `Error` class.

28. d) This program will create file1.txt and file3.txt directories in the root directory, and a file2.txt directory in the "subdir" directory in the root directory.

    The `mkdirs()` method creates a directory for the given name. Since the file names have / in them, the method creates directories in the root directory (or root path for the Windows drive based on the path in which you execute this program).

29. b) When serializing an object that has references to other objects, the serialization mechanism also includes the referenced objects as part of the serialized bytes.
    and
    c) When an object is serialized, the class members that are declared as transient will not be serialized (and hence their values are lost after deserialization).

    Option b) and c) are true regarding object serialization.

    Option a) is wrong because the `Serializable` interface is a marker interface; in other words, the `Serializable` interface is an empty interface and it does not declare any methods in it.
    Option d) is wrong because the `Externalizable` interface declares two methods, `writeExternal()` and `readExternal()`.
    Option e) is wrong because there is no such exception as `NotExternalizableException`.

30. b) Body first head hello program world

    `TreeSet<String>` orders the strings in default alphabetical ascending order and removes duplicates. The delimiter \W is non-word, so the characters such as < act as separators.

31. b) beginIndex = 0 and endIndex = 1

    In the `Path` class's method `subpath(int beginIndex, int endIndex)`, `beginIndex` is the index of the first element (inclusive of that element) and `endIndex` is the index of the last element (exclusive of that element). Note that *the name that is closest to the root in the directory hierarchy has index 0*. Here, the string element `"Program Files"` is the closest to the root `C:\`, so the value of `beginIndex` is 0 and `endIndex` is 1.

32. d) Prints the following: Copy.class Copy.java Hello.class OddEven.class PhotoCopy.java.

    In the Glob pattern "*o*?{java,class,html}", the character * matches any number of characters, so *o* matches any string that has "o" in it. The ? matches exactly one character. The pattern {java,class} matches files with the suffixes of "java" or "class". Hence, from the given files, the matching file names are `Copy.class`, `Copy.java`, `Hello.class`, `OddEven.class`, `PhotoCopy.java`.

33. b) WatchService watcher = FileSystems.getDefault().newWatchService();

    The `getDefault()` method in `FileSystems` returns the reference to the underlying `FileSystem` object. The method `newWatchService()` returns a new watch service that may be used to watch registered objects for changes and events in files or directories.

34. The correct options are

    c) You can get an instance of `PreparedStatement` by calling the `preparedStatement()` method in the `Connection` interface.
    e) The interface `Statement` and its derived interfaces implement the `AutoCloseable` interface, hence they can be used with a try-with-resources statement.

Option c) and e) are correct statements. The other three are incorrect for the following reasons:

- Option a) Objects of type Statement can handle IN, OUT, and INOUT parameters; you need to use objects of CallableStatement type for that.
- Option b) `PreparedStatement` is used for pre-compiled SQL statements; the `CallableStatement` type is used for stored procedures.
- Option d) `CallableStatement` implements the `PreparedStatement` interface; `PreparedStatement` in turn implements the `Statement` interface. These three types are not classes.

35. e) The program will result in throwing a `SQLException` because auto-commit is true.

If you call methods such as `commit()` or `rollback()` when the auto-commit mode is set to true, the program will a `SQLException`.

36. b)
```
RowSetFactory rowSetFactory = RowSetProvider.newFactory();
JdbcRowSet rowSet = rowSetFactory.createJdbcRowSet();
```

37. b) When executed, the program prints `"Worker"` and then the program hangs (i.e., does not terminate).

The statement `Thread.currentThread()` in the `main()` method refers to the "`Master`" thread. Calling the `join()` method on itself means that the thread waits itself to complete, which would never happen, so this program hangs (and does not terminate).

38. Options a) and d) are true:

a) Takes milliseconds as the argument for time to sleep.
d) Can throw the `InterruptedException` if it is interrupted by another thread when it sleeps.

In option b), the `sleep()` method takes milliseconds as an argument, not microseconds.
In option), the `sleep()` method does not relinquish the lock when it goes to sleep; it holds the lock.

39. c) The program prints
Starting to wait
Caught Exception

In this program, the `wait()` method is called without acquiring a lock; hence it will result in throwing an `IllegalMonitorStateException`, which will be caught in the catch block for the `Exception`.

40. a) The program prints
[10, 5, 10, 20]
[20, 5, 10]
[5, 10, 20]
[5, 10, 20]

Here is the description of the containers that explain the output:

- `List` is unsorted.
- `HashSet` is unsorted and retains unique elements.
- `TreeSet` is sorted and retains unique elements.
- `ConcurrentSkipListSet` is sorted and retains unique elements.

41. c) The program does not compile and results in a compiler error in the line marked with comment LINE A.

    The class `CopyOnWriteArrayList` does not inherit from `ArrayList`, so an attempt to assign a `CopyOnWriteArrayList` to an `ArrayList` reference will result in a compiler error (the `ArrayList` suffix in the class named `CopyOnWriteArrayList` could be misleading as these two classes do not share an is-a relationship).

42. c) Prints
    E Gamma
    R Johnson
    R Helm
    J Vlissides
    and then the program terminates.

    The `producer` class puts an author on the list and then sleeps for some time. In the meantime, the other thread (consumer) keeps checking whether the list is non-empty or not. If it is non-empty, the consumer thread removes the item and prints it. Hence, all four author names get printed.

43. b)
    `from=von`
    `subject=betreff`

    In the resource bundle property files, the key values are separated using the = symbol, with each line in the resource file separated by a newline character.

44. c) `ResourceBundle_en.properties`

    Java looks for candidate locales for a base bundle named `ResourceBundle` and locale French (Canada), and checks for the presence of the following property files:

    ```
    ResourceBundle_fr_CA.properties
    ResourceBundle_fr.properties
    ```

    Since both of them are not there, Java searches for candidate locales for the base bundle named `ResourceBundle` and a default locale (English - United States):

    ```
    ResourceBundle_en_US.properties
    ResourceBundle_en.properties
    ```

    Java finds that there is a matching resource bundle, `ResourceBundle_en.properties`. Hence it loads this resource bundle.

45. a) `System.out.println(new SimpleDateFormat("hh:mm:ss").format(new Date()));`

    In the format `hh:mm:ss`, `h` is for the hour in am/pm (with values in 1–12 range), `m` is for minutes, and `s` is for seconds. The class for creating and using custom date or time pattern strings is `SimpleDateFormat`. The expression `new Date()` creates a `Date` object with the current date and time value.

Post-Pretest Evaluation

The 45 questions in this pretest are selected and grouped to represent the 12 topics in the syllabus of the Oracle 1Z0-804 exam. The order of the topics in this pretest replicates the order of the topics in the 1Z0-804 syllabus.

Table 2-1 is your post-pretest evaluation tool. In the rightmost column, fill in the number of your correct pretest answers in each topic from your answer sheet. Wherever your number of correct answers is *less than or equal to* the expected number of correct answers shown in the adjacent column, you will need to focus your preparations for the OPCJP 7 exam on that exam topic and its corresponding chapter in this book.