Mock Exam

The prospect of taking the OCPJP 8 exam raises many questions in your mind.

- "What types of questions are asked in the exam?"
- "What is the format of the questions?"
- "How hard are the questions?"
- "How do I know if I'm ready to take the exam?"

This chapter presents a *mock exam* that helps answer these questions. Use this mock exam as a mental dipstick to gauge how prepared you are to pass the OCPJP 8 exam.

The questions in this mock exam closely mimic the actual questions you will encounter on your OCPJP 8 exam. For instance, you will find these aspects in the actual OCPJP 8 exam: the questions will assume that necessary import statements are included; most questions will contain only relevant code segments (and not complete programs); and questions will appear in random order (and not according to the sequence of exam topics given in the exam syllabus). In this mock exam, we have adopted a similar approach to make this mock exam closely mimic the question format in the actual OCPJP 8 exam.

Before you get started, take a print out of the answer sheet given at the end of this exam. Take this exam as if it were your real OCPJP 8 exam by simulating real test conditions. Find a quiet place where you can take this mock exam without interruption or distraction. Mark your start and finish times, and stop if you cross the exam time limit (2.5 hours). Observe closed-book rules: do not consult the answer key or any other any print, human, or web resources during this mock exam. Check the answers only after you complete the exam. Out of 85 questions, you need to answer at least 55 questions correctly to pass this exam (the passing score is 65%).

Best of luck!

***Time*: 2 hours 30 minutes** ***No. of questions:* 85**

1. **What will be the result of executing this code segment?**

```
Stream.of("ace ", "jack ", "queen ", "king ", "joker ")
        .mapToInt(card -> card.length())
        .filter(len -> len > 3)
        .peek(System.out::print)
        .limit(2);
```

a) This code segment prints: `jack queen king joker`

b) This code segment prints: `jack queen`

c) This code segment prints: `king joker`

d) This code segment does not print anything on the console

2. **Consider the following snippet:**

```
int ch = 0;
try (FileReader inputFile = new FileReader(file)) {
```

```
        // #1
        System.out.print( (char)ch );
    }
}
```

Which one of the following statements can be replaced with statement #1 so that the contents of the file are correctly printed on the console and the program terminates.

a) `while( (ch = inputFile.read()) != null) {`

b) `while( (ch = inputFile.read()) != -1) {`

c) `while( (ch = inputFile.read()) != 0) {`

d) `while( (ch = inputFile.read()) != EOF) {`

3. What will be the output of the following program?

```
class Base {
    public Base() {
        System.out.println("Base");
    }
}
class Derived extends Base {
    public Derived() {
        System.out.println("Derived");
    }
}
class DeriDerived extends Derived {
    public DeriDerived() {
        System.out.println("DeriDerived");
    }
}
class Test {
    public static void main(String []args) {
        Derived b = new DeriDerived();
    }
}
```

a) Base
   Derived
   DeriDerived

b) Derived
   DeriDerived

c) DeriDerived
   Derived
   Base

d) DeriDerived

Derived

e) DeriDerived

4. Given this code segment:

```
final CyclicBarrier barrier =
        new CyclicBarrier(3, () -> System.out.println("Let's
play"));   // LINE_ONE
Runnable r = () -> {                                // LINE_TWO
    System.out.println("Awaiting");
    try {
        barrier.await();
    } catch(Exception e) { /* ignore */ }
};
Thread t1 = new Thread(r);
Thread t2 = new Thread(r);
Thread t3 = new Thread(r);
t1.start();
t2.start();
t3.start();
```

Choose the correct option based on this code segment.

a) This code segment results in a compiler error in line marked with the comment `LINE_ONE`

b) This code segment results in a compiler error in line marked with the comment `LINE_TWO`

c) This code prints:

```
    Let's play
```

d) This code prints:

```
  Awaiting
    Awaiting
    Awaiting
    Let's play
```

e) This code segment does not print anything on the console

5. **Given this class definition:**

```
class Point {
    private int x = 0, y;
    public Point(int x, int y) {
        this.x = x;
```

```
        this.y = y;
    }
    // DEFAULT_CTOR
}
```

Which one of the following definitions of the `Point` constructor can be replaced without compiler errors in place of the comment `DEFAULT_CTOR`?

```
a) public Point() {
        this(0, 0);
        super();
    }
b) public Point() {
        super();
        this(0, 0);
    }
c) private Point() {
        this(0, 0);
    }
d) public Point() {
        this();
    }
e) public Point() {
        this(x, 0);
    }
```

6. Consider the following program:

```
class Base {
    public Base() {
        System.out.print("Base ");
    }
    public Base(String s) {
        System.out.print("Base: " + s);
    }
}
class Derived extends Base {
    public Derived(String s) {
        super();         // Stmt-1
        super(s);        // Stmt-2
        System.out.print("Derived ");
    }
}
class Test {
    public static void main(String []args) {
        Base a = new Derived("Hello ");
```

```
    }
}
```

Select three correct options from the following list:

a) Removing only `Stmt-1` will make the program compilable and it will print the following: `Base Derived`

b) Removing only `Stmt-1` will make the program compilable and it will print the following: `Base: Hello Derived`

c) Removing only `Stmt-2` will make the program compilable and it will print the following: `Base Derived`

d) Removing both `Stmt-1` and `Stmt-2` will make the program compilable and it will print the following: `Base Derived`

e) Removing both `Stmt-1` and `Stmt-2` will make the program compilable and it will print the following: `Base: Hello Derived`

7. Consider the following program and choose the right option from the given list:

```
class Base {
    public void test() {
        protected int a = 10;    // #1
    }
}
class Test extends Base {                // #2
    public static void main(String[] args) {
        System.out.printf(null);        // #3
    }
}
```

a) The compiler will report an error at statement marked with the comment #1

b) The compiler will report an error at statement marked with the comment #2

c) The compiler will report errors at statement marked with the comment #3

d) The program will compile without any error

8. Given this code segment:

```
LocalDate joiningDate = LocalDate.of(2014, Month.SEPTEMBER, 20);
LocalDate now = LocalDate.of(2015, Month.OCTOBER, 20);
// GET_YEARS
System.out.println(years);
```

Which one of the following statements when replaced by the comment GET_YEARS will print 1 on the console?

a) `Period years = Period.between(joiningDate, now).getYears();`

b) `Duration years = Period.between(joiningDate, now).getYears();`

c) `int years = Period.between(joiningDate, now).getYears();`

d) `Instant years = Period.between(joiningDate, now).getYears();`

9. Consider the following program:

```
class Outer {
    class Inner {
        public void print() {
            System.out.println("Inner: print");
        }
    }
}
class Test {
    public static void main(String []args) {
        // Stmt#1
        inner.print();
    }
}
```

Which one of the following statements will you replace with `// Stmt#1` **to make the program compile and run successfully to print "Inner: print" in console?**

a) `Outer.Inner inner = new Outer.Inner();`

b) `Inner inner = new Outer.Inner();`

c) `Outer.Inner inner = new Outer().Inner();`

d) `Outer.Inner inner = new Outer().new Inner();`

10. Consider the following program:

```
public class Outer {
    private int mem = 10;
    class Inner {
        private int imem = new Outer().mem;                // ACCESS1
    }
    public static void main(String []s) {
        System.out.println(new Outer().new Inner().imem); // ACCESS2
    }
}
```

Which one of the following options is correct?

a) When compiled, this program will result in a compiler error in line marked with comment `ACCESS1`

b) When compiled, this program will result in a compiler error in line marked with comment `ACCESS2`

c) When executed, this program prints 10

d) When executed, this program prints 0

11.Consider the following program:

```
interface EnumBase { }
enum AnEnum implements EnumBase {      // IMPLEMENTS_INTERFACE
    ONLY_MEM;
}
class EnumCheck {
    public static void main(String []args) {
        if(AnEnum.ONLY_MEM instanceof AnEnum) {
            System.out.println("yes, instance of AnEnum");
        }
        if(AnEnum.ONLY_MEM instanceof EnumBase) {
            System.out.println("yes, instance of EnumBase");
        }
        if(AnEnum.ONLY_MEM instanceof Enum) {    // THIRD_CHECK
            System.out.println("yes, instance of Enum");
        }
    }
}
```

Which one of the following options is correct?

a) This program results in a compiler error in the line marked with comment `IMPLEMENTS_INTERFACE`

b) This program results in a compiler in the line marked with comment `THIRD_CHECK`

c) When executed, this program prints the following:

```
yes, instance of AnEnum
```

d) When executed, this program prints the following:

```
yes, instance of AnEnum
yes, instance of EnumBase
```

e) When executed, this program prints the following:

```
yes, instance of AnEnum
```

```
yes, instance of EnumBase
yes, instance of Enum
```

12. **Which of the following statements are true with respect to enums? (Select all that apply.)**

a) An `enum` can have `private` constructor

b) An `enum` can have `public` constructor

c) An `enum` can have `public` methods and fields

d) An `enum` can implement an interface

e) An `enum` can extend a class

13. Choose the correct option based on this program:

```
class base1 {
    protected int var;
}
interface base2 {
    int var = 0; // #1
}
class Test extends base1 implements base2 { // #2
    public static void main(String args[]) {
        System.out.println("var:" + var);   // #3
    }
}
```

a) The program will report a compilation error at statement marked with the comment #1

b) The program will report a compilation error at statement marked with the comment #2

c) The program will report a compilation error at statement marked with the comment #3

d) The program will compile without any errors

14. Consider the following program:

```
class WildCard {
    interface BI {}
    interface DI extends BI {}
    interface DDI extends DI {}
    static class C<T> {}
    static void foo(C<? super DI> arg) {}
    public static void main(String []args) {
        foo(new C<BI>());       // ONE
        foo(new C<DI>());       // TWO
        foo(new C<DDI>());      // THREE
```

```
        foo(new C());           // FOUR
    }
}
```

Which of the following options are correct?

a) Line marked with comment `ONE` will result in a compiler error

b) Line marked with comment `TWO` will result in a compiler error

c) Line marked with comment `THREE` will result in a compiler error

d) Line marked with comment `FOUR` will result in a compiler error

15.Consider the following definitions:

```
interface BI {}
interface DI extends BI {}
```

The following options provide definitions of a template class X. Which one of the options specifies class X with a type parameter whose upper bound declares DI to be the super type from which all type arguments must be derived?

a) `class X <T super DI> { }`

b) `class X <T implements DI> { }`

c) `class X <T extends DI> { }`

d) `class X <T extends ? & DI> { }`

16.In the context of Singleton pattern, which one of the following statements is true?

a) A `Singleton` class must not have any static members

b) A `Singleton` class has a public constructor

c) A `Factory` class may use `Singleton` pattern

d) All methods of the `Singleton` class must be private

17.Consider the following program:

```
class ClassA {}
interface InterfaceB {}
class ClassC {}
class Test extends ClassA implements InterfaceB {
    String msg;
    ClassC classC;
}
```

Which one of the following statements is true?

a) Class `Test` is related with `ClassA` with a HAS-A relationship.

b) Class `Test` is related to `ClassC` with a composition relationship.

c) Class `Test` is related with `String` with an IS-A relationship.

d) Class `ClassA` is related with `InterfaceB` with an IS-A relationship.

18.Choose the correct option based on the following code segment:

```
Comparator<String> comparer =
        (country1, country2) ->
                country2.compareTo(country2); // COMPARE_TO
String[] brics = {"Brazil", "Russia", "India", "China"};
Arrays.sort(brics, null);
Arrays.stream(brics).forEach(country -> System.out.print(country
+ " "));
```

a) The program results in a compiler error in the line marked with the comment `COMPARE_TO`

b) The program prints the following: Brazil Russia India China

c) The program prints the following: Brazil China India Russia

d) The program prints the following: Russia India China Brazil

e) The program throws the exception `InvalidComparatorException`

f) The program throws the exception `InvalidCompareException`

g) The program throws the exception `NullPointerException`

19.Which one of the following class definitions will compile without any errors?

```
a)  class P<T> {
        static T s_mem;
    }
b)  class Q<T> {
        T mem;
        public Q(T arg) {
            mem = arg;
        }
    }
c)  class R<T> {
        T mem;
        public R() {
            mem = new T();
        }
    }
d)  class S<T> {
        T []arr;
        public S() {
            arr = new T[10];
        }
    }
```

20.**In a class that extends** `ListResourceBundle`**, which one of the following method definitions correctly overrides the** `getContents()` **method of the base class?**

a)
```
public String[][] getContents() {
        return new Object[][] { { "1", "Uno" }, { "2", "Duo" }, { "3", "Trie" }};
    }
```

b)
```
public Object[][] getContents() {
        return new Object[][] { { "1", "Uno" }, { "2", "Duo" }, { "3", "Trie" }};
    }
```

c)
```
private List<String> getContents() {
        return new ArrayList (Arrays.AsList({ { "1", "Uno" }, { "2", "Duo" }, { "3", "Trie" }});
    }
```

d)
```
protected Object[] getContents(){
        return new String[] { "Uno", "Duo", "Trie" };
    }
```

21.**Which one of the following interfaces declares a single abstract method named** `iterator()`? **(Note: Implementing this interface allows an object to be the target of the for-each statement.)**

a) `Iterable<T>`

b) `Iterator<T>`

c) `Enumeration<E>`

d) `ForEach<T>`

22.Choose the correct option based on this program:

```
import java.util.stream.Stream;
public class Reduce {
    public static void main(String []args) {
        Stream<String> words = Stream.of("one", "two", "three");
        int len = words.mapToInt(String::length).reduce(0, (len1, len2) -> len1 + len2);
        System.out.println(len);
    }
}
```

a) This program does not compile and results in compiler error(s)

b) This program prints: onetwothree

c) This program prints: 11

d) This program throws an `IllegalArgumentException`

23.Which one of the following options is best suited for generating random numbers in a multi-threaded application?

a) Using `java.lang.Math.random()`

b) Using `java.util.concurrent.ThreadLocalRandom`

c) Using `java.util.RandomAccess`

d) Using `java.lang.ThreadLocal<T>`

24.Given this code segment:

```
DateTimeFormatter fromDateFormat
= DateTimeFormatter.ofPattern("MM/dd/yyyy");
// PARSE_DATE
DateTimeFormatter toDateFormat
= DateTimeFormatter.ofPattern("dd/MMM/YY");
System.out.println(firstOct2015.format(toDateFormat));
```

Which one of the following statements when replaced with the comment PARSE_DATE will result in the code to print "`10/Jan/15`"?

a) `DateTimeFormatter firstOct2015 = DateTimeFormatter.parse("01/10/2015", fromDateFormat);`

b) `LocalTime firstOct2015 = LocalTime.parse("01/10/2015", fromDateFormat);`

c) `Period firstOct2015 = Period.parse("01/10/2015", fromDateFormat);`

d) `LocalDate firstOct2015 = LocalDate.parse("01/10/2015", fromDateFormat);`

25.Consider the following program:

```
import java.util.*;
class ListFromVarargs {
    public static <T> List<T> asList1(T... elements) {
        ArrayList<T> temp = new ArrayList<>();
        for(T element : elements) {
            temp.add(element);
        }
        return temp;
    }
    public static <T> List<?> asList2(T... elements) {
        ArrayList<?> temp = new ArrayList<>();
        for(T element : elements) {
            temp.add(element);
        }
        return temp;
```

```
    }
    public static <T> List<?> asList3(T... elements) {
        ArrayList<T> temp = new ArrayList<>();
        for(T element : elements) {
            temp.add(element);
        }
        return temp;
    }
    public static <T> List<?> asList4(T... elements) {
        List<T> temp = new ArrayList<T>();
        for(T element : elements) {
            temp.add(element);
        }
        return temp;
    }
}
```

Which of the `asList` definitions in this program will result in a compiler error?

a) The definition of `asList1` will result in a compiler error

b) The definition of `asList2` will result in a compiler error

c) The definition of `asList3` will result in a compiler error

d) The definition of `asList4` will result in a compiler error

e) None of the definitions (`asList1, asList2, asList3, asList4`) will result in a compiler error

26.Given this code segment:

```
IntFunction<UnaryOperator<Integer>> func = i -> j -> i * j;
// LINE
System.out.println(apply);
```

Which one of these statements when replaced by the comment marked with LINE will print 200?

a) `Integer apply = func.apply(10).apply(20);`

b) `Integer apply = func.apply(10, 20);`

c) `Integer apply = func(10 , 20);`

d) `Integer apply = func(10, 20).apply();`

27.Given this code segment:

```
List<Map<List<Integer>, List<String>>> list = new
ArrayList<>(); // ADD_MAP
Map<List<Integer>, List<String>> map = new HashMap<>();
list.add(null);                                        // ADD_NULL
list.add(map);
```

```
list.add(new HashMap<List<Integer>, List<String>>()); //
ADD_HASHMAP
list.forEach(e ->  System.out.print(e + " "));        // ITERATE
```

Which one of the following options is correct?

a) This program will result in a compiler error in line marked with comment `ADD_MAP`

b) This program will result in a compiler error in line marked with comment `ADD_HASHMAP`

c) This program will result in a compiler error in line marked with comment `ITERATE`

d) When run, this program will crash, throwing a `NullPointerException` in line marked with comment `ADD_NULL`

e) When run, this program will print the following: `null {} {}`

28.Given this code snippet:

```
LocalDate dateOfBirth = LocalDate.of(1988, Month.NOVEMBER, 4);
MonthDay monthDay =
       MonthDay.of(dateOfBirth.getMonth(),
dateOfBirth.getDayOfMonth());
boolean ifTodayBirthday =
       monthDay.equals(MonthDay.from(LocalDate.now())); // COMPARE
System.out.println(ifTodayBirthday ? "Happy birthday!" : "Yet
another day!");
```

Assume that today's date is 4th November 2015. Choose the correct answer based on this code segment.

a) This code will result in a compiler error in the line marked with the comment `COMPARE`

b) When executed, this code will throw `DateTimeException`

c) This code will print: `Happy birthday!`

d) This code will print: `Yet another day!`

29.Consider the following program:

```
class Base<T> { }
class Derived<T> { }
class Test {
    public static void main(String []args) {
        // Stmt #1
    }
}
```

Which statements can be replaced with `// Stmt#1` **and the program remains compilable (choose two):**

a) `Base<Number> b = new Base<Number>();`

b) `Base<Number> b = new Derived<Number>();`

c) `Base<Number> b = new Derived<Integer>();`

d) `Derived<Number> b = new Derived<Integer>();`

e) `Base<Integer> b = new Derived<Integer>();`

f) `Derived<Integer> b = new Derived<Integer>();`

30. **Which of the following classes in the** `java.util.concurrent.atomic` **package inherit from** `java.lang.Number`**? (Select all that apply.)**

a) `AtomicBoolean`

b) `AtomicInteger`

c) `AtomicLong`

d) `AtomicFloat`

e) `AtomicDouble`

31. Given the class definition:

```
class Student{
        public Student(int r) {
                rollNo = r;
        }
        int rollNo;
}
```

Choose the correct option based on this code segment:

```
HashSet<Student> students = new HashSet<>();
students.add(new Student(5));
students.add(new Student(10));
System.out.println(students.contains(new Student(10)));
```

a) This program prints the following: `true`

b) This program prints the following: `false`

c) This program results in compiler error(s)

d) This program throws `NoSuchElementException`

32. Which of the following statements are true regarding resource bundles in the context of localization? (Select ALL that apply.)

a) `java.util.ResourceBundle` is the base class and is an abstraction of resource bundles that contain locale-specific objects

b) `java.util.PropertyResourceBundle` is a concrete subclass of `java.util.ResourceBundle` that manages resources for a locale using strings provided in the form of a property file

c) Classes extending `java.util.PropertyResourceBundle` must override the `getContents()` method which has the return type `Object [][]`

d) `java.util.ListResourceBundle` defines the `getKeys()` method that returns enumeration of keys contained in the resource bundle

33. **Which of the following statements is true regarding the classes or interfaces defined in the** `java.util.concurrent` **package? (Select ALL that apply.)**

a) The `Executor` interface declares a single method `execute(Runnable command)` that executes the given command at sometime in the future

b) The `Callable` interface declares a single method `call()` that computes a result

c) The `CopyOnWriteArrayList` class is not thread-safe unlike `ArrayList` that is thread-safe

d) The `CyclicBarrier` class allows threads to wait for each other to reach a common barrier point

34. **Given these two class declarations:**

```
class CloseableImpl implements Closeable {
    public void close() throws IOException {
        System.out.println("In CloseableImpl.close()");
    }
}
class AutoCloseableImpl implements AutoCloseable {
    public void close() throws Exception {
        System.out.println("In AutoCloseableImpl.close()");
    }
}
```

Choose the correct option based on this code segment:

```
try (Closeable closeableImpl = new CloseableImpl();
        AutoCloseable autoCloseableImpl = new AutoCloseableImpl())
{
} catch (Exception ignore) {
    // do nothing
}
finally {
    // do nothing
}
```

a) This code segment does not print any output in console

b) This code segment prints the following output: `In AutoCloseableImpl.close()`

c) This code segment prints the following output: `In AutoCloseableImpl.close()In CloseableImpl.close()`

d) This code segment prints the following output: `In CloseableImpl.close()In AutoCloseableImpl.close()`

35. **Choose the correct option based on this code segment:**

```
List<Integer> ints = Arrays.asList(1, 2, 3, 4, 5);
ints.replaceAll(i -> i * i); // LINE
System.out.println(ints);
```

a) This code segment prints: `[1, 2, 3, 4, 5]`

b) This program prints: `[1, 4, 9, 16, 25]`

c) This code segment throws `java.lang.UnsupportedOperationException`

d) This code segment results in a compiler error in the line marked with the comment `LINE`

36. **Choose the correct option for this code snippet:**

```
public static void main(String []files) {
    try (FileReader inputFile = new FileReader(new
File(files[0]))) { // #1
            inputFile.close();
  // #2
    }
    catch (FileNotFoundException | IOException e)
{                    // #3
            e.printStackTrace();
    }
}
```

a) The code snippet will compile without any errors

b) The compiler will report an error at statement marked with the comment #1

c) The compiler will report an error at statement marked with the comment #2

d) The compiler will report an error at statement marked with the comment #3

37. Given this program:

```
import java.time.*;
import java.time.temporal.ChronoUnit;
```

```
class DecadeCheck {
    public static void main(String []args) {
        Duration tenYears
= ChronoUnit.YEARS.getDuration().multipliedBy(10);
        Duration aDecade = ChronoUnit.DECADES.getDuration();
        assert tenYears.equals(aDecade) : "10 years is not
a decade!";
    }
}
```

Assume that this program is invoked as follows:

```
java DecadeCheck
```

Choose the correct option based on this program:

a) This program does not compile and results in compiler error(s)

b) When executed, this program prints: `10 years is not a decade!`

c) When executed, this program throws an `AssertionError` with the message "`10 years is not a decade!`"

d) When executed, this program does not print any output and terminates normally

38.Consider the following code segment:

```
while( (ch = inputFile.read()) != VALUE) {
        outputFile.write( (char)ch );
}
```

Assume that `inputFile` **is of type** `FileReader`**, and** `outputFile` **is of type** `FileWriter`**, and** `ch` **is of type** `int`**. The method** `read()` **returns the character if successful, or** `VALUE` **if the end of the stream has been reached. What is the correct value of this** `VALUE` **checked in the** `while` **loop for end-of-stream?**

a) `-1`

b) `0`

c) `255`

d) `Integer.MAX_VALUE`

e) `Integer.MIN_VALUE`

39.Consider the following code snippet.

```
String srcFile = "Hello.txt";
String dstFile = "World.txt";
try (BufferedReader inputFile = new BufferedReader(new
FileReader(srcFile));
     BufferedWriter outputFile = new BufferedWriter(new
FileWriter(dstFile))) {
```

```
    int ch = 0;
    inputFile.skip(6);
    while( (ch = inputFile.read()) != -1) {
        outputFile.write( (char)ch );
    }
    outputFile.flush();
} catch (IOException exception) {
    System.err.println("Error " + exception.getMessage());
}
```

Assume that you have a file named Hello.txt in the current directory with the following contents:

```
Hello World!
```

Which one of the following options correctly describes the behavior of this code segment (assuming that both `srcFile` and `dstFile` are opened successfully)?

a) The program will throw an `IOException` because `skip()` is called before calling `read()`

b) The program will result in creating the file `World.txt` with the contents "World!" in it

c) This program will result in throwing `CannotSkipException`

d) This program will result in throwing `IllegalArgumentException`

40.Consider the following code segment:

```
try (BufferedReader inputFile = new BufferedReader(new
FileReader(srcFile));
    BufferedWriter outputFile
        = new BufferedWriter(new FileWriter(dstFile))) {  // TRY-
BLOCK
    int ch = 0;
    while( (ch = inputFile.read()) != -1) {              // COND-
CHECK
        outputFile.write( (char)ch );
    }
} catch (Exception exception) {
    System.err.println("Error in opening or processing file "
        + exception.getMessage());
}
```

Assume that `srcFile` and `dstFile` are Strings. Choose the correct option.

a) This program will get into an infinite loop because the condition check for end-of-stream `(checking != -1)` is incorrect

b) This program will get into an infinite loop because the variable `ch` is declared as `int` instead of `char`

c) This program will result in a compiler error in line marked with comment TRY-BLOCK because you need to use , (comma) instead of ; (semi-colon) as separator for opening multiple resources

d) This program works fine and copies `srcFile` to `dstFile`

41.Given the following definitions:

```
interface InterfaceOne<T> {
    void foo();
}
interface InterfaceTwo<T> {
    T foo();
}
interface InterfaceThree<T> {
    void foo(T arg);
}
interface InterfaceFour<T> {
    T foo(T arg);
}
public class DateLambda {
    public static void main(String []args) {
        // STATEMENT
        System.out.println(val.foo());
    }
}
```

Which one of the following statements can be replaced with the line marked with the comment STATEMENT that the program will print the result that is same as the call `LocalDateTime.now()`?

a) `InterfaceOne<LocalDateTime> val = LocalDateTime::now;`

b) `InterfaceTwo<LocalDateTime> val = LocalDateTime::now;`

c) `InterfaceThree<LocalDateTime> val = LocalDateTime::now;`

d) `InterfaceFour<LocalDateTime> val = LocalDateTime::now;`

42.Which one of the following statements will compile without errors?

a) `Locale locale1 = new Locale.US;`

b) `Locale locale2 = Locale.US;`

c) `Locale locale3 = new US.Locale();`

d) `Locale locale4 = Locale("US");`

e) `Locale locale5 = new Locale(Locale.US);`

43.Choose the correct option based on this code segment:

```
String []exams = { "OCAJP 8", "OCPJP 8", "Upgrade to OCPJP 8" };
Predicate isOCPExam = exam -> exam.contains("OCP");
// LINE-1
List<String> ocpExams = Arrays.stream(exams)
                              .filter(exam -> exam.contains("OCP"))
                              .collect(Collectors.toList());      //
LINE-2
boolean result =
       ocpExams.stream().anyMatch(exam ->
exam.contains("OCA")); // LINE-3
System.out.println(result);
```

a) This code results in a compiler error in line marked with the comment `LINE-1`

b) This code results in a compiler error in line marked with the comment `LINE-2`

c) This code results in a compiler error in line marked with the comment `LINE-3`

d) This program prints: `true`

e) This program prints: `false`

44.Which one of the following code snippets shows the correct usage of try-with-resources statement?

```
a)  public static void main(String []files) {
       try (FileReader inputFile
              = new FileReader(new File(files[0]))) {
                  //...
       }
       catch(IOException ioe) {}
    }
b)  public static void main(String []files) {
       try (FileReader inputFile
              = new FileReader(new File(files[0]))) {
                  //...
       }
       finally { }
       catch(IOException ioe) {}
    }
c)  public static void main(String []files) {
       try (FileReader inputFile
              = new FileReader(new File(files[0]))) {
                  //...
       }
```

```
        catch(IOException ioe) {}
        finally { inputFile.close(); }
    }
```

d)
```
    public static void main(String []files) {
        try (FileReader inputFile
                = new FileReader(new File(files[0]))) {
                    //...
        }
    }
```

45.Two friends are waiting for some more friends to come so that they can go to a restaurant for dinner together. Which synchronization construct could be used here to programmatically simulate this situation?

a) `java.util.concurrent.RecursiveTask`

b) `java.util.concurrent.locks.Lock`

c) `java.util.concurrent.CyclicBarrier`

d) `java.util.concurrent.RecursiveAction`

46.Choose the correct option based on this program:

```
import java.util.*;
public class ResourceBundle_it_IT extends ListResourceBundle {
    public Object[][] getContents() {
        return contents;
    }
    static final Object[][] contents = {
                { "1", "Uno" },
                { "2", "Duo" },
                { "3", "Trie" },
    };
    public static void main(String args[]) {
        ResourceBundle resBundle =
                ResourceBundle.getBundle("ResourceBundle", new
Locale("it", "IT", ""));
        System.out.println(resBundle.getObject(new
Integer(1).toString()));
    }
}
```

a) This program prints the following: `Uno`

b) This program prints the following: `1`

c) This program will throw a `MissingResourceException`

d) This program will throw a `ClassCastException`

47.Given this code segment:

```
Set<String> set = new CopyOnWriteArraySet<String>(); // #1
set.add("2");
set.add("1");
Iterator<String> iter = set.iterator();
set.add("3");
set.add("-1");
while(iter.hasNext()) {
    System.out.print(iter.next() + " ");
}
```

Choose the correct option based on this code segment.

a) This code segment prints the following: 2 1

b) This code segment the following: 1 2

c) This code segment prints the following: -1 1 2 3

d) This code segment prints the following: 2 1 3 -1

e) This code segment throws a `ConcurrentModificationException`

f) This code segment results in a compiler error in statement #1

48.Choose the correct option based on this code segment:

```
Stream<Integer> ints = Stream.of(1, 2, 3, 4);
boolean result
= ints.parallel().map(Function.identity()).isParallel();
System.out.println(result);
```

a) This code segment results in compiler error(s)

b) This code segment throws `InvalidParallelizationException` for the call `parallel()`

c) This code segment prints: `false`

d) This code segment prints: `true`

49.Choose the correct option based on this code segment:

```
Path currPath = Paths.get(".");
try (DirectoryStream<Path> javaFiles
= Files.newDirectoryStream(currPath, "*.{java}")) {
    for(Path javaFile : javaFiles) {
        System.out.println(javaFile);
    }
} catch (IOException ioe) {
    System.err.println("IO Error occurred");
    System.exit(-1);
}
```

a) This code segment throws a `PatternSyntaxException`

b) This code segment throws an `UnsupportedOperationException`

c) This code segment throws an `InvalidArgumentException`

d) This code segment lists the files ending with suffix `.java` in the current directory

50.Given this code segment:

```
Path aFilePath = Paths.get("D:\\dir\\file.txt");
Iterator<Path> paths = aFilePath.iterator();
while(paths.hasNext()) {
    System.out.print(paths.next() + " ");
}
```

Choose the correct option assuming that you are using a Windows machine and the file `D:\dir\file.txt` does not exist in the underlying file system.

a) The program throws a `FileNotFoundException`

b) The program throws an `InvalidPathException`

c) The program throws an `UnsupportedOperationException`

d) The program gets into an infinite loop and keeps printing: `path element: dir`

e) The program prints the following: `dir file.txt`

51.Which of the following is NOT a problem associated with thread synchronization using mutexes?

a) Deadlock

b) Lock starvation

c) Type erasure

d) Livelock

52.Assume that a thread acquires a lock on an object obj; the same thread again attempts to acquire the lock on the same object obj. What will happen?

a) If a thread attempts to acquire a lock again, it will result in throwing an `IllegalMonitorStateException`

b) If a thread attempts to acquire a lock again, it will result in throwing an `AlreadyLockAcquiredException`

c) It is okay for a thread to acquire lock on `obj` again, and such an attempt will succeed

d) If a thread attempts to acquire a lock again, it will result in a deadlock

53.Which one of the following interfaces is empty (i.e., an interface that does not declare any methods)?

a) `java.lang.AutoCloseable` interface

b) `java.util.concurrent.Callable<T>` interface

c) `java.lang.Cloneable` interface

d) `java.lang.Comparator<T>` interface

54.Consider the following program and choose the correct option that describes its output:

```
import java.util.concurrent.atomic.AtomicInteger;
class Increment {
    public static void main(String []args) {
        AtomicInteger i = new AtomicInteger(0);
        increment(i);
        System.out.println(i);
    }
    static void increment(AtomicInteger atomicInt){
        atomicInt.incrementAndGet();
    }
}
```

a) 0

b) 1

c) This program throws an `UnsafeIncrementException`

d) This program throws a `NonThreadContextException`

55.What is the output of the following program?

```
class EnumTest {
    enum Directions { North, East, West, South };
    enum Cards { Spade, Hearts, Club, Diamond };
    public static void main(String []args) {
        System.out.println("equals: " +
Directions.East.equals(Cards.Hearts));
        System.out.println("ordinals: " +
                        (Directions.East.ordinal() ==
Cards.Hearts.ordinal()));
    }
}
```

a)

```
    equals: false
    ordinals: false
```

b)

```
    equals: true
    ordinals: false
```

c)

```
    equals: false
```

```
    ordinals: true
```

d)

```
    equals: true
    ordinals: true
```

56.Consider the following program and choose the correct option:

```
import java.util.concurrent.atomic.AtomicInteger;
class AtomicVariableTest {
    private static AtomicInteger counter = new AtomicInteger(0);
    static class Decrementer extends Thread {
        public void run() {
            counter.decrementAndGet(); // #1
        }
    }
    static class Incrementer extends Thread {
        public void run() {
            counter.incrementAndGet(); // #2
        }
    }
    public static void main(String []args) {
        for(int i = 0; i < 5; i++) {
            new Incrementer().start();
            new Decrementer().start();
        }
        System.out.println(counter);
    }
}
```

a) This program will always print 0

b) This program will print any value between -5 to 5

c) If you make the `run()` methods in the `Incrementer` and `Decrementer` classes synchronized, this program will always print 0

d) The program will report compilation errors at statements #1 and #2

57.Which one of the following statements will compile without any errors?

a) `Supplier<LocalDate> now = LocalDate::now();`

b) `Supplier<LocalDate> now = () -> LocalDate::now;`

c) `String now = LocalDate::now::toString;`

d) `Supplier<LocalDate> now = LocalDate::now;`

58.For the following enumeration definition, which one of the following prints the value 2 in the console?

```
enum Pets { Cat, Dog, Parrot, Chameleon };
```

a) `System.out.print(Pets.Parrot.ordinal());`

b) `System.out.print(Pets.Parrot);`

c) `System.out.print(Pets.indexAt("Parrot"));`

d) `System.out.print(Pets.Parrot.value());`

e) `System.out.print(Pets.Parrot.getInteger());`

59.Assume that the current directory is "D:\workspace\ch14-test".

Choose the correct option based on this code segment:

```
Path testFilePath = Paths.get(".\\Test");
System.out.println("file name:" + testFilePath.getFileName());
System.out.println("absolute path:"
+ testFilePath.toAbsolutePath());
System.out.println("Normalized path:" + testFilePath.normalize());
```

a) `file name:Test`
`absolute path:D:\workspace\ch14-test\.\Test`
`Normalized path:Test`

b) `file name:Test`
`absolute path:D:\workspace\ch14-test\Test`
`Normalized path:Test`

c) `file name:Test`
`absolute path:D:\workspace\ch14-test\.\Test`
`Normalized path:D:\workspace\ch14-test\.\Test`

d) `file name:Test`
`absolute path:D:\workspace\ch14-test\.\Test`
`Normalized path:D:\workspace\ch14-test\Test`

60.Given this code segment:

```
BufferedReader br = new BufferedReader(new
FileReader("names.txt"));
System.out.println(br.readLine());
br.mark(100);   // MARK
System.out.println(br.readLine());
br.reset();     // RESET
System.out.println(br.readLine());
```

Assume that names.txt exists in the current directory, and opening the file succeeds, and br points to a valid object. The content of the names.txt is the following:

```
olivea
emma
margaret
emily
```

Choose the correct option.

a) This code segment prints the following:

```
olivea
emma
margaret
```

b) This code segment prints the following:

```
olivea
emma
olivea
```

c) This code segment prints the following:

```
olivea
emma
emma
```

d) This code segment throws an `IllegalArgumentException` in the line `MARK`

e) This code segment throws a `CannotResetToMarkPositionException` in the line `RESET`

61. **Given this class definition:**

```
abstract class Base {
    public abstract Number getValue();
}
```

Which of the following two options are correct concrete classes extending `Base` class?

a)

```
    class Deri extends Base {
        protected Number getValue() {
            return new Integer(10);
        }
    }
```

b)

```
    class Deri extends Base {
        public Integer getValue() {
            return new Integer(10);
        }
    }
```

c)

```
    class Deri extends Base {
        public Float getValue(float flt) {
            return new Float(flt);
        }
```

```
    }
```

d)

```
    class Deri extends Base {
        public java.util.concurrent.atomic.AtomicInteger getValue()
{
            return new
java.util.concurrent.atomic.AtomicInteger(10);
        }
    }
```

62. **Which TWO of the following classes are defined in the** `java.util.concurrent.atomic` **package?**

a) `AtomicBoolean`

b) `AtomicDouble`

c) `AtomicReference<V>`

d) `AtomicString`

e) `AtomicObject<V>`

63. **Given the following class and interface definitions:**

```
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 "); }
}
```

Select the correct option for this code segment:

```
Penguin pingu = new Penguin();
pingu.walk();
pingu.fly();
```

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"

64.Given this class definition:

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

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?

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

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

b) `Outer.new Inner().text`

c) `Outer.Inner.text`

d) `new Outer().Inner.text`

65.Given this code snippet:

```
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 the underlying file system has the 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.)

Choose the correct option:

a) This code segment 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 code segment 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 code segment 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 code segment 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

66.Given these class definitions:

```
class Book {
    public void read() {
        System.out.println("read!");
    }
}
public class BookUse {
    // DEFINE READBOOK HERE
    public static void main(String []args) {
        new BookUse().readBook(Book::new);
    }
}
```

Which one of the following code segments when replaced with the comment "DEFINE READBOOK HERE"inside the BookUse class will result in printing "read!" on the console?

a)

```
    private void readBook(Supplier<? extends Book> book) {
        book.get().read();
    }
```

b)

```
    private static void readBook(Supplier<? extends Book> book) {
        Book::read;
    }
```

c)

```
    private void readBook(Consumer<? extends Book> book) {
        book.accept();
    }
```

d)

```
    private void readBook(Function<? extends Book> book) {
        book.apply(Book::read);
    }
```

67.Given the class definition:

```
class Employee {
    String firstName;
    String lastName;
    public Employee (String fName, String lName) {
        firstName = fName;
        lastName = lName;
    }
    public String toString() { return firstName + " " +
lastName; }
    String getFirstName() { return firstName; }
    String getLastName() { return lastName; }
}
```

Here is a code segment:

```
Employee[] employees = { new Employee("Dan", "Abrams"),
                         new Employee("Steve", "Nash"),
                         new Employee("John", "Nash"),
                         new Employee("Dan", "Lennon"),
                         new Employee("Steve", "Lennon")
                       };
Comparator<Employee> sortByFirstName =
                    ((e1, e2) ->
e1.getFirstName().compareTo(e2.getFirstName()));
Comparator<Employee> sortByLastName =
                    ((e1, e2) ->
e1.getLastName().compareTo(e2.getLastName()));
// SORT
```

The sorting needs to be performed in descending order of the first names; when first names are the same, the names should then be sorted in ascending order of the last names. For that, which one of the following code segment will you replace for the line marked by the comment `SORT`?

a) `Stream.of(employees)`

```
   .sorted(sortByFirstName.thenComparing(sortByLastName))
   .forEach(System.out::println);
```

b) `Stream.of(employees)`

```
   .sorted(sortByFirstName.reversed().thenComparing(sortByLastName
))
   .forEach(System.out::println);
```

c) `Stream.of(employees)`

```
    .sorted(sortByFirstName.thenComparing(sortByLastName).reversed(
))
    .forEach(System.out::println);
```

d) `Stream.of(employees)`

```
    .sorted(sortByFirstName.reversed().thenComparing(sortByLastName
).reversed())
    .forEach(System.out::println);
```

68.Given this code snippet:

```
Statement statement = connection.createStatement
        (ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet resultSet = statement.executeQuery
        ("SELECT * FROM EMPLOYEE WHERE EMPNAME = \"John\"");
resultSet.updateString("EMPNAME", "Jonathan");
// UPDATE
```

Assume that the variable connection points to a valid `Connection` **object and there exists an employee record with** `EMPNAME` **value "John". The** `resultSet` **is updated by changing the value of** `EMPNAME` **column with the value "Jonathan" instead of "John". For this change to be reflected in the underlying database, which one of the following statements will you replace with the comment** `UPDATE`**?**

a) `connection.updateAllResultSets();`

b) `resultSet.updateRow();`

c) `statement.updateDB();`

d) `connection.updateDatabase();`

69.**Given these class definitions:**

```
class ReadDevice implements AutoCloseable {
    public void read() throws Exception {
        System.out.print("read; ");
        throw new Exception();
    }
    public void close() throws Exception {
        System.out.print("closing ReadDevice; ");
    }
}
class WriteDevice implements AutoCloseable {
```

```
    public void write() {
        System.out.print("write; ");
    }
    public void close() throws Exception {
        System.out.print("closing WriteDevice; ");
    }
}
```

What will this code segment print?

```
try(ReadDevice rd = new ReadDevice();
    WriteDevice wd = new WriteDevice()) {
    rd.read();
    wd.write();
} catch(Exception e) {
    System.out.print("Caught exception; ");
}
```

a) `read; closing WriteDevice; closing ReadDevice; Caught exception;`

b) `read; write; closing WriteDevice; closing ReadDevice; Caught exception;`

c) `read; write; closing ReadDevice; closing WriteDevice; Caught exception;`

d) `read; write; Caught exception; closing ReadDevice; closing WriteDevice;`

e) `read; Caught exception; closing ReadDevice; closing WriteDevice;`

70. **Select all the statements that are true about streams (supported in `java.util.stream.Stream` interface)?**

a) Computation on source data is performed in a stream only when the terminal operation is initiated, i.e., streams are "lazy"

b) Once a terminal operation is invoked on a stream, it is considered consumed and cannot be used again

c) Once a stream is created as a sequential stream, its execution mode cannot be changed to parallel stream (and vice versa)

d) If the stream source is modified when the computation in the stream is being performed, then it may result in unpredictable or erroneous results

71. Given the code segment:

```
List<Integer> integers = Arrays.asList(15, 5, 10, 20, 25, 0);
// GETMAX
```

Which of the code segments can be replaced for the comment marked with GETMAX to return the maximum value?

a) `Integer max = integers.stream().max((i, j) -> i - j).get();`

b) `Integer max = integers.stream().max().get();`

c) `Integer max = integers.max();`

d) `Integer max = integers.stream().mapToInt(i -> i).max();`

72. **Given the class definition:**

```
class NullableBook {
    Optional<String> bookName;
    public NullableBook(Optional<String> name) {
        bookName = name;
    }
    public Optional<String> getName() {
        return bookName;
    }
}
```

Choose the correct option based on this code segment:

```
NullableBook nullBook = new
NullableBook(Optional.ofNullable(null));
Optional<String> name = nullBook.getName();
name.ifPresent(System.out::println).orElse("Empty"); // NULL
```

a) This code segment will crash by throwing `NullPointerException`

b) This code segment will print: `Empty`

c) This code segment will print: `null`

d) This code segment will result in a compiler error in line marked with `NULL`

73. **Choose the correct option for this code segment:**

```
List<String> lines = Arrays.asList("foo;bar;baz", "", "qux;norf");
lines.stream()
      .flatMap(line -> Arrays.stream(line.split(";"))) // FLAT
      .forEach(str -> System.out.print(str + ":"));
```

a) This code will result in a compiler error in line marked with the comment `FLAT`

b) This code will throw a `java.lang.NullPointerException`

c) This code will throw a `java.util.regex.PatternSyntaxException`

d) This code will print `foo:bar:baz::qux:norf:`

74. Choose the correct option based on this code segment:

```
LocalDate feb28th = LocalDate.of(2015, Month.FEBRUARY, 28);
System.out.println(feb28th.plusDays(1));
```

a) This program prints: `2015-02-29`

b) This program prints: `2015-03-01`

c) This program throws a `java.time.DateTimeException`

d) This program throws

a `java.time.temporal.UnsupportedTemporalTypeException`

75.**Choose the correct option based on this code segment:**

```
List<Integer> ints = Arrays.asList(1, 2, 3, 4, 5);
ints.removeIf(i -> (i % 2 ==0)); // LINE
System.out.println(ints);
```

a) This code segment prints: `[1, 3, 5]`

b) This code segment prints: `[2, 4]`

c) This code segment prints: `[1, 2, 3, 4, 5]`

d) This code segment throws `java.lang.UnsupportedOperationException`

e) This code segment results in a compiler error in the line marked with the comment `LINE`

76.Given the class definition:

```
class Point {
        public int x, y;
        public Point(int x, int y) {
                this.x = x;
                this.y = y;
        }
        public int getX() { return x; }
        public int getY() { return y; }
        // other methods elided
}
```

Which one of the following enforces encapsulation? (Select all that apply.)

a) Make data members `x` and `y` private

b) Make the `Point` class public

c) Make the constructor of the `Point` class private

d) Remove the getter methods `getX()` and `getY()` methods from the `Point` class

e) Make the `Point` class static

77.Given the definition:

```
class Sum implements Callable<Long> {    // LINE_DEF
    long n;
    public Sum(long n) {
        this.n = n;
```

```
    }
    public Long call() throws Exception {
        long sum = 0;
        for(long longVal = 1; longVal <= n; longVal++) {
            sum += longVal;
        }
        return sum;
    }
}
```

Given that the sum of 1 to 5 is 15, select the correct option for this code segment:

```
Callable<Long> task = new Sum(5);
ExecutorService es = Executors.newSingleThreadExecutor(); //
LINE_FACTORY
Future<Long> future = es.submit(task);                    //
LINE_CALL
System.out.printf("sum of 1 to 5 is %d", future.get());
es.shutdown();
```

a) This code results in a compiler error in the line marked with the comment `LINE_DEF`

b) This code results in a compiler error in the line marked with the comment `LINE_FACTORY`

c) This code results in a compiler error in the line marked with the comment `LINE_CALL`

d) This code prints: `sum of 1 to 5 is 15`

78.Given this class definition:

```
public class AssertCheck {
    public static void main(String[] args) {
        int score = 0;
        int num = 0;
        assert ++num > 0 : "failed";
        int res = score / num;
        System.out.println(res);
    }
}
```

Choose the correct option assuming that this program is invoked as follows:

```
java -ea AssertCheck
```

a) This program crashes by throwing `java.lang.AssertionError` with the message "`failed`"

b) This program crashes by throwing `java.lang.ArithmeticException` with the message "`/ by zero`"

c) This program prints: `0`

d) This program prints "`failed`" and terminates normally

79.Given this code segment:

```
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
String integer = br.readLine();
// CODE
System.out.println(val);
```

Which one of the following statements when replaced by the comment CODE will successfully read an integer value from console?

a) `int val = integer.getInteger();`

b) `int val = Integer.parseInt(integer);`

c) `int val = String.parseInt(integer);`

d) `int val = Number.parseInteger(integer);`

80.**Which one of the following definitions of the** `AResource` **class implementation is correct so that it can be used with try-with-resources statement?**

a)
```
class AResource implements Closeable {
        protected void close() /* throws IOException */ {
            // body of close to release the resource
        }
    }
```

b)
```
class AResource implements Closeable {
        public void autoClose() /* throws IOException */ {
            // body of close to release the resource
        }
    }
```

c)
```
class AResource implements AutoCloseable {
        void close() /* throws IOException */ {
            // body of close to release the resource
        }
    }
```

d)
```
class AResource implements AutoCloseable {
        public void close() throws IOException {
            // body of close to release the resource
        }
    }
```

81. **Which of the following are functional interfaces? (Select all that apply.)**

a)
```
@FunctionalInterface
  interface Foo {
    void execute();
  }
```

b)
```
@FunctionalInterface
  interface Foo {
    void execute();
    boolean equals(Object arg0);
  }
```

c)
```
@FunctionalInterface
  interface Foo {
    boolean equals(Object arg0);
  }
```

d)
```
interface Foo{}
```

82. **Choose the correct option based on this code segment:**

```
Stream<String> words = Stream.of("eeny", "meeny", "miny",
"mo");          // LINE_ONE
String boxedString = words.collect(Collectors.joining(", ", "[",
"]"));  // LINE_TWO
System.out.println(boxedString);
```

a) This code results in a compiler error in line marked with the comment `LINE_ONE`

b) This code results in a compiler error in line marked with the comment `LINE_TWO`

c) This program prints: `[eeny, meeny, miny, mo]`

d) This program prints: `[eeny], [meeny], [miny], [mo]`

83. **Choose the correct option based on the following code snippet. Assume that** `DbConnector.connectToDb()` **returns a valid** `Connection` **object and that the** `EMPLOYEE` **table has a column named** `CUSTOMERID` **of type** `VARCHAR(3)`**.**

```
ResultSet resultSet = null;
try (Connection connection = DbConnector.connectToDb())
{            // LINE_ONE
    Statement statement = connection.createStatement();
    resultSet = statement.executeQuery
```

```
                ("SELECT * FROM CUSTOMER WHERE CUSTOMERID = 1212");
// LINE_TWO
}
while (resultSet.next()){      // LINE_THREE
    resultSet.getString("CUSTOMERID");
}
```

a) This code results in a compiler error in line marked with the comment `LINE_ONE`

b) This code results in a compiler error in line marked with the comment `LINE_TWO`

c) This code results in a compiler error in line marked with the comment `LINE_THREE`

d) This code prints `"1212"` on the console and terminates

e) This code gets into an infinite loop and keeps printing `"1212"` on the console

f) This code throws `SQLException`

84.Given this code snippet:

```
public static Connection connectToDb() throws SQLException {
    String url = "jdbc:mysql://localhost:3306/";
    String database = "addressBook";
    String userName = "root";
    String password = "mysql123";
    // CONNECT_TO_DB
}
```

Which one of the following statements will you replace with the comment `CONNECT_TO_DB` to create a `Connection` object?

a) `return DatabaseManager.getConnection(url, database, userName, password);`

b) `return Connection.getConnection(url, database, userName, password);`

c) `return DriverManager.getConnection(url + database, userName, password);`

d) `return DatabaseDriver.getConnection(url + database, userName, password);`

85.Choose the correct option based on this code segment:

```
Path path = Paths.get("file.txt");
// READ_FILE
lines.forEach(System.out::println);
```

Assume that a file named "file.txt" exists in the directory in which this code segment is run and has the content "hello". Which one of these options can be replaced by the text READ_FILE that will successfully read the "file.txt" and print "hello" on the console?

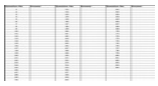
a) `List<String> lines = Files.lines(path);`

b) `Stream<String> lines = Files.lines(path);`

c) `Stream<String> lines = File.readLines(path);`

d) `Stream<String> lines = Files.readAllLines(path);`

Answer Sheet



Answers and Explanations

1. d) This code segment does not print anything on the console
The `limit()` method is an intermediate operation and not a terminal operation. Since there is no terminal operation in this code segment, elements are not processed in the stream and hence it does not print anything on the console.

2. b) `while( (ch = inputFile.read()) != -1) {`
The `read()` method returns -1 when the file reaches the end.
Why other options are wrong:
Option a) Since `ch` is of type `int`, it cannot be compared with `null`.
Option c) With the check != 0, the program will never terminate since `inputFile.read()` returns -1 when it reaches end of the file.
Option d) Using the identifier `EOF` will result in a compiler error.

3. a) `Base`
  `Derived`
  `DeriDerived`
Whenever a class gets instantiated, the constructor of its base classes (the constructor of the root of the hierarchy gets executed first) gets invoked before the constructor of the instantiated class.

4. d) This code prints:
`Awaiting`

`Awaiting`
`Awaiting`
`Let's play`

There are three threads expected in the `CyclicBarrier` because of the value 3 given as the first argument to the`CyclicBarrier` constructor. When a thread executes, it prints "`Awaiting`" and awaits for the other threads (if any) to join. Once all three threads join, they cross the barrier and the message "`Let's play`" gets printed on the console.

5.

c) `private Point() {`
`this(0, 0);`
`}`

Options a) and b) Both the calls `super()` and `this()` cannot be provided in a constructor

Option d) The call `this();` will result in a recursive constructor invocation for `Point()` constructor (and hence the compiler will issue an error)

Option e) You cannot refer to an instance field `x` while explicitly invoking a constructor using `this` keyword

6.

b) Removing `Stmt-1` will make the program compilable and it will print the following: Base: Hello Derived.

c) Removing `Stmt-2` will make the program compilable and it will print the following: `Base Derived`

d) Removing both `Stmt-1` and `Stmt-2` will make the program compilable and it will print the following: `Base Derived`

Why other options are wrong:

Option a) If you remove `Stmt`-1, a call to `super`(s) will result in printing `Base: Hello`, and then constructor of the`Derived` class invocation will print `Derived`. Hence it does not print: `Base Derived`.

Option e) If you remove `Stmt-1` and `Stmt-2`, you will get a compilable program but it will result in printing: `Base Derived`and not `Base: Hello Derived`.

7.

a) The compiler will report an error at statement line marked with the comment #1

Statement #1 will result in a compiler error since the keyword `protected` is not allowed inside a method body. You cannot provide access specifiers (`public`, `protected`, or `private`) inside a method body.

Why other options are wrong:

Option b) It is acceptable to extend a base class and hence there is no compiler error in line marked with comment #2.

Option c) It is acceptable to pass null to `printf` function hence there is no compiler error in line marked with comment #2.

Option d) This program will not compile cleanly and hence this option is wrong.

8.

c) `int years = Period.between(joiningDate, now).getYears();`

The `between()` method in `Period` returns a `Period` object.

The `getYears()` method called on the returned `Period`returns an `int`. Hence, option c) that declares years as `int` is the correct option.

Using the other three options will result in compiler errors because the `getYears()` method of `Period` return an `int`.

9.

d) `Outer.Inner inner = new Outer().new Inner();`

Option d) uses the correct syntax for instantiating Outer and Inner classes.

The other three options will result in compiler error(s).

10.

c) This program runs and prints 10

An inner class can access even the private members of the outer class.

Similarly, the private variable belonging to the inner class can be accessed in the outer class.

Why other options are wrong:

Options a) and b) are wrong because this program compiles without any errors. The variable `mem` is initialized to value 10 and that gets printed by the program (and not 0) and hence Option d) is wrong.

11.

e) When executed, this program prints the following:

```
yes, instance of AnEnum
yes, instance of EnumBase
yes, instance of Enum
```

An enumeration can implement an interface (but cannot extend a class, or cannot be a base class). Each enumeration constant is an object of its enumeration type. An enumeration automatically extends the abstract class `java.util.Enum`. Hence, all the three `instanceof` checks succeed.
Why other options are wrong:
This program compiles cleanly and hence options a) and b) are wrong.
Options c) and d) do not provide the complete output of the program and hence they are also incorrect.

12.

a) An `enum` can have `private` constructor
c) An `enum` can have `public` methods and fields
d) An `enum` can implement an interface
Why other options are wrong:
Option b) An `enum` cannot have `public` constructor(s)
Option e) An `enum` cannot extend a class

13.

c) The program will report a compilation error at statement marked with the comment #3
Statements marked with the comment #1 and #2 will not result in any compiler errors; only access to the variable `var`will generate a compiler error since the access is ambiguous (since the variable is declared in both `base1` and `base2`).

14.

c) The line marked with comment `THREE` will result in a compiler error
Options a) and b) For the substitution to succeed, the type substituted for the wildcard `?` should be `DI` or one of its super types.
Option c) The type `DDI` is not a super type of `DI`, so it results in a compiler error.
Option d) The type argument is not provided, meaning that C is a raw type in the expression `new C()`. Hence, this will elicit a compiler warning, but not an error.

15.

c) `class X <T extends DI> { }`
The keyword `extends` is used to specify the upper bound for type `T`; with this, only the classes or interfaces implementing the interface `DI` can be used as a

replacement for `T`. Note that the `extends` keyword is used for any base type—irrespective of whether the base type is a class or an interface.

16.

c) A `Factory` class may use `Singleton` pattern

A `Factory` class generates the desired type of objects on demand. Hence, it might be required that only one `Factory`object exists; in this case, `Singleton` can be employed in a `Factory` class.

Why other options are wrong:

a) A `Singleton` class needs to have a static member to return a singleton instance

b) A `Singleton` class must declare its constructor(s) private to ensure that they are not instantiated

d) A static method (typically named `getInstance()`) with public access may need to be provided to get the instance of the `Singleton` class.

17.

b) Class `Test` is related with `ClassC` with a composition relationship.

When a class inherits from another class, they share an IS-A relationship. On the other hand, if a class uses another class (by declaring an instance of another class), then the first class has a HAS-A relationship with the used class.

18.

c) The program prints the following: Brazil China India Russia.

For the `sort()` method, `null` value is passed as the second argument, which indicates that the elements' "natural ordering" should be used. In this case, natural ordering for `Strings` results in the strings sorted in ascending order. Note that passing `null` to the `sort()` method does not result in a `NullPointerException`.

The statement marked with `COMPARE_TO` will compile without errors. Note that the variable `comparer` is unused in this code segment.

19.

```
b) class Q<T> {
        T mem;
        public Q(T arg) {
            mem = arg;
        }
    }
```

Option a) You cannot make a static reference of type `T` in a generic class.

Option c) and d) You cannot instantiate the type `T` or `T[]` using `new` operator in a generic class.

20.

b) `public Object[][] getContents() {`

```
        return new Object[][] { { "1", "Uno" }, { "2", "Duo" }, { "3", "Trie" }};
    }
```

The `getContents()` method is declared in `ListResourceBundle` as follows:

`protected abstract Object[][] getContents()`

The other three definitions are incorrect overrides and will result in compiler error(s).

21.

a) `Iterable<T>`

The interface `Iterable<T>` declares this single method:

`Iterator<T> iterator();`

This `iterator`() method returns an object of type `Iterator`<T>. A class must implement `Iterable`<T> for using its object in a `for-each` loop.

Though `Iterable` interface (in Java 8) defines `forEach()` and `spliterator()` methods, they are default methods and not static methods.

Why other options are wrong:

Option b) The `Iterator<T>` interface declares abstract methods `hasNext()` and `next()`, and defines default methods`remove()` and `forEachRemaining()`.

Option c) The `Enumeration<T>` interface declares hasMoreElements() and `nextElement()` methods.

Option d) There is no interface named `ForEach<T>` in the Java core library.

22.

c) This program prints: 11

This program compiles without any errors. The variable words point to a stream of `String`s. The call`mapToInt(String::length)` results in a stream of `Integer`s with the length of the strings. One of the overloaded versions of `reduce()` method takes two arguments:

`T reduce(T identity, BinaryOperator<T> accumulator);`

The first argument is the `identity` value, which is given as the value 0 here. The second operand is a `BinaryOperator`match for the lambda expression `(len1, len2) -> len1 + len2`. The `reduce()` method thus adds the length of all the three strings in the stream, which results in the value 11.

23.

b) Using `java.util.concurrent.ThreadLocalRandom`
`java.lang.Math.random()` is not efficient for concurrent programs.
Using `ThreadLocalRandom` results in less overhead and contention when compared to using `Random` objects in concurrent programs (and hence using this class type is the best option in this case).
`java.util.RandomAccess` is unrelated to random number generation. This interface is the base interface for random access data structures and is implemented by classes such as `Vector` and `ArrayList`.
`java.lang.ThreadLocal<T>`class provides support for creating thread-local variables.

24.

d) `LocalDate firstOct2015 = LocalDate.parse("01/10/2015", fromDateFormat);`
You need to use `LocalDate` for parsing the date string given in the `DateTimeFormatter` variable `fromDateFormat` (with the format string `MM/dd/yyyy`”). Other options will not compile.

25.

b) The definition of `asList2` will result in a compiler error.
In the `asList2` method definition, `temp` is declared as `ArrayList<?>`. Since the template type is a wild-card, you cannot put any element (or modify the container). Hence, the method call `temp.add(element);` will result in a compiler error.

26.

a) `Integer apply = func.apply(10).apply(20);`
The `IntFunction<R>` takes an argument of type `int` and returns a value of type `R`. The `UnaryOperator<T>` takes an argument of type `T` and returns a value of type `T`.
The correct way to invoke `func` is to call `func.apply(10).apply(10)` (the other three options do not compile). The first call `apply(10)` results in an `Integer` object that is passed to the lambda expression;

calling `apply(20)` results in executing the expression `(i * j)` that evaluates to 200.

The other three options will result in compiler error(s).

27.

e) When run, this program will print the following: `null {} {}`

The lines marked with comments `ADD_MAP` and `ADD_HASHMAP` are valid uses of the diamond operator to infer type arguments. Calling the `add()` method passing `null` does not result in a `NullPointerException`. The program, when run, will successfully print the output `null {} {}` (null output indicates a null value was added to the list, and the {} output indicates that `Map` is empty).

28.

c) This code will print: `Happy birthday!`

This code gets the month-and-day components from the given `LocalDate` and creates a `MonthDay` object. Another way to create a `MonthDay` object is to call the `from()` method and pass a `LocalDate` object. The `equals()` method compares if the month and date components are equal and if so returns true. Since the month and day components are equal in this code (assuming that the today's date is 4th November 2015 as given in the question), it results in printing "`Happy birthday!`".

29.

a) `Base<Number> b = new Base<Number>();`

f) `Derived<Integer> b = new Derived<Integer>();`

Note that `Base` and `Derived` are not related by an inheritance relationship. Further, for generic type parameters, subtyping doesn't work: you cannot assign a derived generic type parameter to a base type parameter.

30.

b) `AtomicInteger`

c) `AtomicLong`

Classes `AtomicInteger` and `AtomicLong` extend `Number` class.

Why other options are wrong:

Option a) `AtomicBoolean` does not extend `java.lang.Number`.

Options d) and e) Classes named as `AtomicFloat` or `AtomicDouble` do not exist in the `java.util.concurrent.atomic`package.

31.

b) This program prints the following: `false`

Since methods `equals()` and `hashcode()` methods are not overridden in the `Student` class, the `contains()` method will not work as intended and prints false.

32.

a) `ResourceBundle` is the base class and is an abstraction of resource bundles that contain locale-specific objects

b) `java.util.PropertyResourceBundle` is a concrete subclass of `java.util.ResourceBundle` that manages resources for a locale using strings provided in the form of a property file

d) `java.util.ListResourceBundle` defines the `getKeys()` method that returns enumeration of keys contained in the resource bundle

The option c) is not to be selected. There is no such method named `getContents()` method that has the return type`Object [][]`. It has the method `getKeys()` that returns an enumeration of keys contained in the resource bundle. It is classes that extend `java.util.ListResourceBundle` (and not `java.util.PropertyResourceBundle` as given in this option) that must override the `getContents()` method that has the return type `Object [][]`.

33.

a) The `Executor` interface declares a single method `execute(Runnable command)` that executes the given command at some time in the future

b) The `Callable` interface declares a single method `call()` that computes a result

d) The `CyclicBarrier` class allows threads to wait for each other to reach a common barrier point

These three options are true.

Option c) is incorrect because the `CopyOnWriteArrayList` class is thread-safe whereas `ArrayList` class is not thread-safe.

34.

c) This code segment prints the following output:

```
In AutoCloseableImpl.close()
In CloseableImpl.close()
```

The types implementing `AutoCloseable` can be used with a try-with-resources statement. The `Closeable` interface extends `AutoCloseable`, so

classes implementing `Closeable` can also be used with a try-with-resources statement.

The `close()` methods are called in the opposite order when compared to the order of resources acquired in the try-with-resources statement. So, this program calls the `close()` method of `AutoCloseableImpl` first, and after that calls the`close()` method on the `CloseableImpl` object.

35.

b) This program prints: `[1, 4, 9, 16, 25]`

The `replaceAll()` method (added in Java 8 to the `List` interface) takes an `UnaryOperator` as the argument. In this case, the unary operator squares the integer values. Hence, the program prints [1, 4, 9, 16, 25]. The underlying `List`object returned by `Arrays.asList()` method can be modified using the `replaceAll()` method and it does not result in throwing `java.lang.UnsupportedOperationException`.

36.

d) The compiler will report an error at the statement marked with the comment #3

Both of the specified exceptions belong to the same hierarchy (`FileNotFoundException` derives from an `IOException`), so you cannot specify both exceptions together in the multi-catch handler block.

It is not a compiler error to explicitly call `close()` method for a `FileReader` object inside a try-with-resources block.

37.

d) When executed, this program does not print any output and terminates normally

The program compiles cleanly without any errors. Assertions are disabled by default. Since assertions are not enabled when invoking this program, it does not evaluate the `assert` expression. Hence, the program terminates normally without printing any output on the console.

38.

a) -1

The `read()` method returns the value -1 if end-of-stream (EOS) is reached, which is checked in this `while` loop.

39.

b) The program will result in creating the file `World.txt` with the contents "World!" in it.

The method call `skip(n)` skips `n` bytes (i.e., moves the buffer pointer by `n` bytes). In this case, 6 bytes need to be skipped, so the string "Hello" is not copied in the `while` loop while reading and writing the file contents.

Why other options are wrong:

Option a) The `skip()` method can be called before the `read()` method.

Option c) No exception named `CannotSkipException` exists.

Option d) The `skip()` method will throw an `IllegalArgumentException` only if a negative value is passed.

40.

d) This program works fine and copies `srcFile` to `dstFile`

Why other options are wrong:

Options a) and b) This program does not get into an infinite loop because the condition check for end-of-stream `(checking != -1)` is correct and the variable `ch` needs to be declared as `int` (and not `char`).

Option c) You can use ; (semi-colon) as separator for opening multiple resources in try-with-resources statement.

41.

b) `InterfaceTwo<LocalDateTime> val = LocalDateTime::now;`

The method `now()` in `LocalDateTime` is declared with the signature: `LocalDateTime now()`

The matching functional interface should also have an abstract method that takes no argument and returns a value of type `T`. Since `InterfaceTwo` has the abstract method declared as `T foo()`, the statement`InterfaceTwo<LocalDateTime> val = LocalDateTime::now;` succeeds. From the interface, the method can be invoked with `val.foo();` since `val` refers to `LocalDateTime::now`, and it is equivalent to making the call `LocalDateTime.now()`.

42.

b) `Locale locale2 = Locale.US;`

The `static public final Locale US` member in the `Locale` class is accessed using the expression `Locale.US`, as in option b).

The other options will result in compiler error(s).

43.

a) This code results in a compiler error in line marked with the comment `LINE-1`

The functional interface `Predicate<T>` takes type `T` as the generic parameter that is not specified in `LINE-1`. This results in a compiler error because the lambda expression uses the method `contains()` in the call `exam.contains("OCP")`.

If `Predicate<String>` were specified (as in `Predicate isOCPExam = exam -> exam.contains("OCP")`), this code segment would compile without errors, and when executed will print "false".

44.

a)
```
public static void main(String []files) {
        try (FileReader inputFile
            = new FileReader(new File(files[0]))) {
            //...
        }
        catch(IOException ioe) {}
    }
```

Why other options are wrong:

- Option b) provides finally before the catch block, it will result in a compiler error.
- Option c) uses the variable `inputFile` in the statement `inputFile.close()` that is not accessible in the finally block and hence results in a compiler error. Option d) the required `catch` block in this context is missing in the code (because the try block code may throw `IOException`), and hence it is incorrect usage.

45.

c) `java.util.concurrent.CyclicBarrier`

`CyclicBarrier` is used when threads may need to wait at a predefined execution point until all other threads reach that point. This construct matches the given requirements.

Why other options are wrong:

- Options a) and d) `java.util.concurrent.RecursiveTask` and `java.util.concurrent`

.RecursiveAction are used in the context of executing tasks in fork-join framework.

- Option b) The java.util.concurrent.locks.Lock class provides better abstraction for locking and unlocking than using the synchronized keyword.

46.

a) This program prints the following: Uno

This program correctly extends ListResourceBundle and defines a resource bundle for the locale it_IT.

The getObject() method takes String as an argument; this method returns the value of the matching key. The expression new Integer(1).toString() is equivalent of providing the key "1", so the program prints Uno in the console.

47.

a) This code segment prints the following: 2 1

This code segment modifies the underlying CopyOnWriteArrayList container object using the add() method. After adding the elements "2" and "1", the iterator object is created. After creating this iterator object, two more elements are added, so internally a copy of the underlying container is created due to this modification to the container. But the iterator still refers to the original container that had two elements. So, this program results in printing 2 and 1. If a new iterator is created after adding these four elements, it will iterate over all those four elements.

48.

d) This code segment prints: true

The stream pointed by ints is a sequential stream because sequential is the default execution mode. The call toparallel() method changes the execution mode to parallel stream. The isParallel() method returns true because the current execution mode of the stream is parallel.

Why other options are wrong:

Option a) This code compiles without errors. The call to map(Function.identity()) is acceptable because the argumentFunction.identity() just returns the same stream element it is passed with.

Option b) It is possible to change the execution mode of a stream after it is created, and it does not result in throwing any exceptions.

Option c) The `isParallel()` method returns the current execution mode and not the execution mode when the stream was created. So the `isParallel()` method returns true in this code (and not false as given in this option).

49.

d) This code segment lists the files ending with suffix `.java` in the current directory

The path "." specifies the current directory. The pattern "`*.{java}`" matches file names with suffix `.java`.

50.

e) This code segment prints the following: `dir file.txt`

The name elements in a `path` object are identified based on the separators. Note: To iterate name elements of the `Path`object does not actually require that the corresponding files/directories must exist, so it will not result in throwing any exceptions.

51.

c) Type erasure

Deadlocks, lock starvation, and livelocks are problems that arise when using mutexes for thread synchronization. Type erasure is a concept related to generics where the generic type information is lost once the generic type is compiled.

52.

c) It is okay for a thread to acquire lock on `obj` again, and such an attempt will succeed

Java locks are reentrant: a Java thread, if it has already acquired a lock, can acquire it again, and such an attempt will succeed. No exception is thrown and no deadlock occurs for this case.

53.

c) `java.lang.Cloneable` interface

From the documentation of `clone()` method: "By *convention*, classes that implement this interface should override the`Object.clone()` method. Note that this interface does *not* contain the `clone` method."

Why other options are wrong:

- Option a) The `AutoCloseable` interface declares the `close()` method.
- Option b) `Callable` declares `call()` method.
- Option d) The `Comparator<T>` interface declares `compare()` and `equals()` methods.

54.

b) `1`

The call `atomicInt.incrementAndGet();` mutates the integer value passed through the reference variable `atomicInt`, so the changed value is printed in the `main()` method. Note that `AtomicInteger` can be used in thread or non-thread context though it is not of any practical use when used in single-threaded programs.

55.

c)

```
equals: false
ordinals: true
```

The `equals()` method returns true only if the enumeration constants are the same. In this case, the enumeration constants belong to different enumerations, so the `equals()` method returns `false`. However, the ordinal values of the enumeration constants are equal since both are second elements in their respective enumerations.

56.

b) This program will print any value between −5 to 5

You have employed `AtomicInteger`, which provides a set of `atomic` methods such as `incrementAndGet()` and`decrementAndGet()`. Hence, you will always get 0 as the final value of counter. However, depending on thread scheduling, the intermediate counter values may be anywhere between −5 to +5, Hence the output of the program can range between −5 and +5.

57.

d) `Supplier<LocalDate> now = LocalDate::now;`

The `now()` method defined in `LocalDate` does not take any arguments and returns a `LocalDate` object. Hence, the signature of `now()` method matches that of the only abstract method in the `Supplier` interface: `T get()`. Hence, the method reference `Local::now` can be assigned to `Supplier<LocalDate>` and the statement compiles without any errors.

Other options show improper use of method reference and they will result in compiler error(s).

58.

a) `System.out.print(Pets.Parrot.ordinal());`

The `ordinal()` method prints the position of the enumeration constant within an enumeration and hence it prints 2 for this program.

Why other options are wrong:

- Option b) The call `print(Pets.Parrot);` prints the string "Parrot" to console
- Options c), d) and e) There are no methods named `indexAt()`, `value()`, or `getInteger()` in `Enum`

59.

a)

```
file name:Test
absolute path:D:\workspace\ch14-test\.\Test
Normalized path:Test
```

The absolute path adds the path from the root directory; however, it does not normalize the path. Hence, "`.\`" will be retained in the resultant path. On the other hand, the `normalize`() method normalizes the path but does not make it absolute.

60.

c) This code segment prints the following:

```
olivea
emma
emma
```

The method `void mark(int limit)` in `BufferedReader` marks the current position for resetting the stream to the marked position. The argument limit specifies the number of characters that may be read while still preserving the mark. This code segment marks the position after "`olivea`" is read, so after reading "`emma`," when the marker is reset and the line is read again, it reads "`emma`" once again.

61.

b)

```
    class Deri extends Base {
        public Integer getValue() {
            return new Integer(10);
        }
```

```
}
```

d)

```
    class Deri extends Base {
        public java.util.concurrent.atomic.AtomicInteger getValue()
{
            return new
java.util.concurrent.atomic.AtomicInteger(10);
        }
    }
```

Option b) makes use of a co-variant return type (note that `Integer` extends `Number`), and defines the overriding method correctly.

Option d) makes use of co-variant return type (note that `AtomicInteger` extends `Number`), and defines the overriding method correctly.

Why the other two options are wrong:

- Option a) attempts to assign a weaker access privilege by declaring the method protected when the base method is public, and thus is incorrect (results in a compiler error).
- In option c) the method `Float getValue(float flt)` does not override the `getValue`() method in `Base` since the signature does not match, so it is incorrect (results in a compiler error).

62.

a) `AtomicBoolean` and c) `AtomicReference<V>`

The class `AtomicBoolean` supports atomically updatable Boolean values. The class `AtomicReference`<V> supports atomically updatable references of type `V`. Classes `AtomicDouble`, `AtomicString`, and `AtomicObject` are not part of the`java.util.concurrent.atomic` package.

63.

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).

64.

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.

65.

d) This code segment 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).

66.

a)

```
    private void readBook(Supplier<? extends Book> book) {
        book.get().read();
    }
```

The `Supplier<T>` interface declares the abstract method `get()`.

The `get()` method does not take any arguments and returns an object of type `T`. Hence, the call `book.get().read()` succeeds and prints "read!" on the console.

Why other options are wrong:

- Option b) Method references can be used in places where lambda expressions can be used. Hence, this code segment will result in a compiler error.
- Option c) The `accept()` method in the `Consumer<T>` interface requires an argument to be passed – since it is missing here, it will result in a compiler error.
- Option d) The `Function<T, R>` interface takes two type parameters and hence this method definition will result in a compiler error.

67.

b)

```
    Stream.of(employees)
            .sorted(sortByFirstName.reversed().thenComparing(sortBy
LastName))
            .forEach(System.out::println);
```

The `sortByFirstName` is a `Comparator` that sorts names by the `Employee`'s first name. Because we need to sort the names in descending order, we need to call the `reversed()` method. After that, we need to sort the last names in ascending order, and hence we can call `thenComparing(sortByLastName)`.

68.

b) `resultSet.updateRow();`

The call `updateRow()` on the `ResultSet` object updates the database. Other options will not compile.

69.

a) `read; closing WriteDevice; closing ReadDevice; Caught exception;`

The `read()` method of `ReadDevice` throws an exception, and hence the `write()` method of `WriteDevice` is not called. The try-with-resources statement releases the resources in the reverse order from which they were acquired. Hence, the`close()` for `WriteDevice` is called first, followed by the call to the `close()` method for `ReadDevice`. Finally, the `catch`block prints "`Caught exception;`" to the console.

70.

a) Computation on source data is performed in a stream only when the terminal operation is initiated, i.e., streams are "lazy"

b) Once a terminal operation is invoked on a stream, it is considered consumed and cannot be used again

d) If the stream source is modified when the computation in the stream is being performed, then it may result in unpredictable or erroneous results

These three statements are true about streams.

Option c) is not correct. Once a stream is created as a sequential its execution mode can be changed to parallel stream by calling `parallel()` method. Similarly, once a parallel stream is created, you can make it a sequential stream by calling `sequential()` method.

71.

a) `Integer max = integers.stream().max((i, j) -> i - j).get();`

Calling `stream()` method on a `List<Integer>` object results in a stream of type `Stream<Integer>`. The `max()` method takes a `Comparator` as the argument that is provided by the lambda expression `(i, j) -> i - j`.

The `max()` method returns an `Optional<Integer>` and the `get()` method returns an `Integer` value.

Why other options are wrong:

- Option b) The `max()` method in `Stream` requires a `Comparator` to be passed as the argument
- Option c) There is no `max()` method in `List<Integer>`
- Option d) The `mapToInt()` method returns an `IntStream`, but the `max()` method returns an `OptionalInt` and hence it cannot be assigned to `Integer` (as required in this context)

72.

d) This program will result in a compiler error in line marked with `NULL`

The `ifPresent()` method for `Optional` takes a `Consumer` as the argument and returns `void`. Hence, it is not possible to chain the `orElse()` method after calling the `ifPresent()` method.

73.

d) This code will print `foo:bar:baz::qux:norf:`

The `flatMap()` method flattens the streams by taking the elements in the stream. The elements in the given strings are split using the separator ";" and the elements from the resulting string stream are collected.

The `forEach()` method prints the resulting strings.

Why other options are wrong:

- Option a) This code does not issue any compiler errors
- Option a) This code does not issue any compiler errorsOption b) This Splitting an empty string does not result in a null, and hence this code does not throw `NullPointerException.`
- Option a) This code does not issue any compiler errorsOption c) The syntax of the given regular expression is correct and hence it does not result in `PatternSyntaxException.`

74.

b) This program prints: `2015-03-01`

Since 2015 is not a leap year, there are only 28 days in February. Hence adding a day from 28th February 2015 results in 1st March 2015 and that is printed.

75.

d) This code segment throws `java.lang.UnsupportedOperationException`

The underlying `List` object returned by `Arrays.asList()` method is a fixed-size list and hence we cannot remove elements from that list. Hence calling `removeIf()` method on this list results in throwing an`UnsupportedOperationException`.

76.

a) Make data members `x` and `y` private

Publicly visible data members violate encapsulation since any code can modify the `x` and `y` values of a `Point` object directly. It is important to make data members private to enforce encapsulation.

Why other options are wrong:

- Options b), c), and d) Making the `Point` class public, making the constructor of the class private or removing the getter methods will not help enforce encapsulation.
- Option e) You cannot declare a class static.

77.

d) This code prints: `sum of 1 to 5 is 15`

This code correctly uses `Callable<T>`, `ExecutorService`, and `Future<T>` interfaces and the `Executors` class to calculate the sum of numbers from 1 to 5.

78.

c) This program prints: 0

The condition within the `assert` statement `++num > 0` holds true because `num`'s value is 1 with the pre-increment expression `++num`. The expression `0 / 1` results in the value 0 and hence the output.

Why other options are wrong:

- Options a) and d) The assertion condition holds true; hence neither `java.lang.AssertionError` is thrown nor the message "failed" get printed
- Since the assertions are enabled by passing the option "-ea" this does not result in divide-by-zero. If the assertion were not disabled, it would have crashed by throwing `java.lang.ArithmeticException` with the message "`/ by zero`"

79.

b) `int val = Integer.parseInt(integer);`

Using the method `Integer.parseInt(String)` is the correct way to get an `int` value from a `String` object. The other three options will not compile.

80.

d)

```
    class AResource implements AutoCloseable {
        public void close() throws IOException {
            // body of close to release the resource
        }
    }
```

`AutoCloseable` is the base interface of the `Closeable` interface; `AutoCloseable` declares close as `void close() throws Exception;` In `Closeable`, it is declared as `public void close() throws IOException;`. For a class to be used with try-with-resources, it should both implement `Closeable` or `AutoCloseable` and correctly override the `close()` method.

Option a) declares `close()` protected; since the `close()` method is declared public in the base interface, you cannot reduce its visibility to protected, so this will result in a compiler error.

Option b) declares `autoClose();` a correct implementation would define the `close()` method.

Option c) declares `close()` with default access; since the close method is declared public in the base interface, you cannot reduce its visibility to default accesses, so it will result in a compiler error.

Option d) is a correct implementation of the `AResource` class that overrides the `close()` method.

81.

a)

```
    @FunctionalInterface
     interface Foo {
        void execute();
    }
```

b)

```
    @FunctionalInterface
     interface Foo {
        void execute();
        boolean equals(Object arg0);
    }
```

The interface in option a) declares exactly one abstract method and hence it is a functional interface. In option b) note that `equals()` method belongs to `Object` class, which is not counted as an abstract method required for a functional interface. Hence, the interface in option b) has only one abstract method and it qualifies as a functional interface.

Why other options are wrong:

- Option c) the interface does not have an abstract method declared and hence it is not a functional interface.
- Option d) the interface does not have any methods and hence it is not a functional interface.

82.

c) This program prints: `[eeny, meeny, miny, mo]`

`Stream.of()` method takes a variable length argument list of type `T` and it returns a `Stream<T>`. The `joining()` method in `Collectors` class takes `delimiter`, `prefix`, and `suffix` as arguments:

```
joining(CharSequence delimiter, CharSequence prefix, CharSequence suffix)
```

Hence, the expression `Collectors.joining(", ", "[", "]")` joins the strings with commas and encloses the resulting string within '[' and ']'.

83.

f) This code throws `SQLException`

The try-with-resources block is closed before the while statement executes. Hence, call `resultSet.next()` results in making a call on the closed `ResultSet` object, thereby throwing an `SQLException`.

84.

c) `return DriverManager.getConnection(url + database, userName, password);`

The `getConnection()` method in `DriverManager` takes three `String` arguments and returns a `Connection`:

```
Connection getConnection(String url, String user, String password)
```

Hence, option c) is the correct answer.

The other three options will result in compiler errors.

85.

b) `Stream<String> lines = Files.lines(path);`

The `lines(Path)` method in `Files` class takes a `Path` and returns `Stream<String>`. Hence option b) is the correct answer.

Option a) The code segment results in a compiler error because the return type of `lines()` method is `Stream<String>`and not `List<String>`.

Option c) There is no such method named `readLines(Path)` in `Files` that returns a `Stream<String>` and hence it results in a compiler error.

Option d) The `readAllLines(Path)` method returns a `List<String>` and not `Stream<String>` and hence the given statement results in a compiler error.