1z0-829

[Total Questions: 50]

Question #:1

Given:

```
public class Test {
   public static void main(String[] args) {
       List<String>  elements =
             Arrays.asList("car", "truck", "car",
                           "bicycle", "car", "truck", "motorcycle");
       Map<String, Long>  outcome =
             elements.stream().collect(Collectors.groupingBy(Function.identity(), Collectors.counting() ) );
       System.out.println(outcome);
   }
}
```

What is the result?

A. Bicycle =7, car=7, motorcycle=7, truck=7)

B. (3:bicycle, 0:car, 0motercycle, 5:truck)

C. (Bicycle, car, motorcycle, truck)

D. Bicycle-1, car=3, motorcycle=1, truck=2)

E. Compilation fails.

Answer: E

Explanation

The answer is E because the code fragment contains several syntax errors that prevent it from compiling. Some of the errors are:

- The enum declaration is missing a semicolon after the list of constants.
- The enum constants are not capitalized, which violates the Java naming convention for enums.
- The switch expression is missing parentheses around the variable name.
- The case labels are missing colons after the enum constants.
- The default label is missing a break statement, which causes a fall-through to the next case.
- The println statement is missing a closing parenthesis and a semicolon.

A possible corrected version of the code fragment is:

enum Vehicle { BICYCLE, CAR, MOTORCYCLE, TRUCK; } public class Test { public static void

main(String[] args) { Vehicle v = Vehicle.BICYCLE; switch (v) { case BICYCLE: System.out.print(“1”); break; case CAR: System.out.print(“3”); break; case MOTORCYCLE: System.out.print(“1”); break; case TRUCK: System.out.print(“2”); break; default: System.out.print(“0”); break; } System.out.println(); } }

This would print 1 as the output. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Enum Types
- The switch Statement

Question #:2

Given:

```
public class Test {
   public static void main(String[] args) {
      final int x = 2;
      int y = x;
      while (y<3) {
         switch (y) {
            case 0+x:
            y++;
            case 1:
            y++;
         }
      }
      System.out.println(y);
   }
}
```


What is the result?

A. 4

B. 2

C. 6

D. Nothing is printed because of an indefinite loop.

E. Compilation fails.

F. 5

G. A runtime exception is thrown.

H. 3

Answer: E

Explanation

The code will not compile because the variable ‘x’ is declared as final and then it is being modified in the switch statement. This is not allowed in Java. A final variable is a variable whose value cannot be changed once it is initialized1. The switch statement tries to assign different values to ‘x’ depending on the value of ‘y’, which violates the final modifier. The compiler will report an error: The final local variable x cannot be assigned. It must be blank and not using a compound assignment. References: The final Keyword (The Java™ Tutorials > Learning the Java Language > Classes and Objects)

Question #:3

Given:

```
final class Folder {   // line n1
    // line n2
    public void open(){
        System.out.print("Open ");
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        try (Folder f = new Folder()) {
            f.open();
        }
    }
}
```


Which two modifications enable the code to print Open Close?

A)

At line n2, insert:

```
final void close() {
    System.out.print("Close ");
```

B)

Replace line n1 with:

```
class Folder extends Closeable {
```

C)

Replace line n1 with:

```
class Folder extends Exception {
```

D)

Replace line n1 with:

```
class Folder implements AutoCloseable {
```

E)

At line n2, insert:

```
public void close() throws IOException {
    System.out.print("Close ");
}
```


A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

Answer: B E

Explanation

The code given is a try-with-resources statement that declares a resource of type AutoCloseable. The resource is an anonymous class that implements the AutoCloseable interface and overrides the close() method. The code also has a print() method that prints the value of the variable s. The code is supposed to print “Open Close”, but it does not compile because of two errors.

The first error is at line n1, where the anonymous class is missing a semicolon at the end of its declaration. This causes a syntax error and prevents the code from compiling. To fix this error, option B adds a semicolon after the closing curly brace of the anonymous class.

The second error is at line n2, where the print() method is called without an object reference. This causes a compilation error because the print() method is not static and cannot be invoked without an object. To fix this error, option E adds an object reference to the print() method by using the variable t.

Therefore, options B and E are correct and enable the code to print “Open Close”.

Question #:4

Given:

```
class StockException extends Exception {
    public StockException(String s) { super(s); }
}
class OutofStockException extends StockException {
    public OutofStockException(String s) { super(s); }
}
```

and the code fragment:

```
public class Test {
    public static void main(String[] args) throws OutofStockException {
        m();
    }
    public static void m() throws OutofStockException {
        try {
            throw new StockException("Raised.");
        } catch (Exception e) {
            throw new OutofStockException(e.getMessage());
        }
    }
}
```


Which statement is true?

A. The program throws StockException.

B. The program fails to compile.

C. The program throws outofStockException.

D. The program throws ClassCastException

Answer: B

Explanation

The answer is B because the code fragment contains a syntax error that prevents it from compiling. The code fragment tries to catch a StockException in line 10, but the catch block does not have a parameter of type StockException. The catch block should have a parameter of type StockException, such as:

catch (StockException e) { // handle the exception }

This is required by the Java syntax for the catch clause, which must have a parameter that is a subclass of Throwable. Without a parameter, the catch block is invalid and causes a compilation error.

Option A is incorrect because the program does not throw a StockException, as it does not compile.

Option C is incorrect because the program does not throw an OutofStockException, as it does not compile.

Option D is incorrect because the program does not throw a ClassCastException, as it does not compile.
References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- The try-with-resources Statement (The Java™ Tutorials > Essential Classes > Exceptions)
- The catch Blocks (The Java™ Tutorials > Essential Classes > Exceptions)

Question #:5

Assuming that the data, txt file exists and has the following content:

Text1

Text2

Text3

Given the code fragment:

```
try {
    Path p = new File("data.txt").toPath();
    Stream lines = Files.lines(p);
    String data = lines.collect(Collectors.joining("-"));
    System.out.println(data);
    String data2 = Files.readAllLines(p).get(3);
    System.out.println(data2);
} catch (IOException ex) {
    System.out.println(ex);
}
```

What is the result?

A. text1-

text2-

text3-

text3

B. text1-text2-text3

text1

text2

text3

C. text1-text2-text3

A java.lang.indexoutofBoundsException is thrown.

D. text1-text2-text3

text3

Answer: D

Explanation

The answer is D because the code fragment reads the file “data.txt” and collects all the lines in the file into a single string, separated by hyphens. Then, it prints the resulting string. Next, it attempts to read the fourth line in the file (index 3) and print it. However, since the file only has three lines, an IndexOutOfBoundsException is thrown. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Read contents of a file using Files class in Java

Question #:6

Given:

```
public class Test {
   static interface Animal {
   }

   static class Dog implements Animal {
   }

   private static void play(Animal a) {
      System.out.print("flips");
   }

   private static void play(Dog d) {
      System.out.print("runs");
   }

   public static void main(String[] args) {
      Animal a1 = new Dog();
      Dog a2 = new Dog();
      play(a1);
      play(a2);
   }
}
```


What is the result?

A. flipsflips

B. Compilation fails

C. flipsruns

D. runsflips

E. runsruns

Answer: B

Explanation

The code fragment will fail to compile because the play method in the Dog class is declared as private, which means that it cannot be accessed from outside the class. The main method is trying to call the play method on a Dog object, which is not allowed. Therefore, the code fragment will produce a compilation error.

Question #:7

Given:

```
public class App{
    String name;
    public App(String name){
        this.name = name;
    }
    public static void main(String args[]) {
        App t1= new App("t1");
        App t2= new App("t2");
        t1 = t2;
        t1 = null;
        System.out.println("GC");
    }
}
```


Which statement is true while the program prints GC?

A. Only the object referenced by t2 is eligible for garbage collection.

B. Both the objects previously referenced by t1 are eligible for garbage collection.

C. None of the objects are eligible for garbage collection.

D. Only one of the objects previously referenced by t1 is eligible for garbage collection.

Answer: B

Question #:8

Given the code fragment:

```
String s = "10_00";
Integer s2 = 10_00;
// Line n1
System.out.println(res);
```

Which two statements at Line nl independently enable you to print 1250?

A. Integer res = 250 + integer.parseint (s)

B. Integer res = 250 + s:

C. Integer res = 250 + integer (s2):

D. Integer res= 250 + s2;

E. Integer res = 250 + integer . valueof (s);

F. Integer res = 250;

Res = + s2;

Answer: A E

Explanation

The code fragment is creating a string variable “s” with the value “10_00” and an integer variable “s2” with the value 10. The string “s” is using an underscore as a separator for readability, which is allowed in Java SE 171. The question is asking for two statements that can add 250 to the numeric value of “s” and assign it to an integer variable “res”. The correct answers are A and E because they use the methods parseInt and valueOf of the Integer class to convert the string “s” to an integer. Both methods interpret the string as a signed decimal integer and return the equivalent int or Integer value23. The other options are incorrect because they either use invalid syntax, such as B and C, or they do not convert the string “s” to an integer, such as D and F. References: Binary Literals (The Java™ Tutorials > Learning the Java Language > Numbers and Strings), Integer (Java SE 17 & JDK 17), Integer (Java SE 17 & JDK 17)

Question #:9

Given:

```
class Product {
    String name; double price;
    Product(String s, double d) {
        this.name = s;
        this.price = d;
    }
}
class ElectricProduct extends Product {
    ElectricProduct(String name, double price) {
        super(name, price);
    }
}
```

and the code fragment:

```
List<Product> p = List.of(
new ElectricProduct("CellPhone",100),
new ElectricProduct("ToyCar",90),
new ElectricProduct("Motor",200),
new ElectricProduct("Fan",300)
);

DoubleSummaryStatistics sts = p.stream().filter(a -> a instanceof ElectricProduct)
                                .collect(Collectors.summarizingDouble(a ->
a.price));
String s1 = p.stream().filter(a -> a instanceof Product)
                      .collect(Collectors.mapping(p2 -> p2.name, Collectors.joining(",")));
System.out.println(sts.getMax());
System.out.println(s1);
```


A. 300.00

CellPhone,ToyCar,Motor,Fan

B. 100.00

CellPhone,ToyCar,Motor,Fan

C. 100.00 CellPhone,ToyCar

D. 300.00

CellPhone.ToyCar

Answer: A

Explanation

The code fragment is using the Stream API to perform a reduction operation on a list of ElectricProduct objects. The reduction operation consists of three parts: an identity value, an accumulator function, and a combiner function. The identity value is the initial value of the result, which is 0.0 in this case. The accumulator function is a BiFunction that takes two arguments: the current result and the current element of the stream, and returns a new result. In this case, the accumulator function is (a,b) -> a + b.getPrice (), which means that it adds the price of each element to the current result. The combiner function is a BinaryOperator that takes two partial results and combines them into one. In this case, the combiner function is (a,b) -> a + b, which means that it adds the two partial results together.

The code fragment then applies a filter operation on the stream, which returns a new stream that contains only the elements that match the given predicate. The predicate is p -> p.getPrice () > 10, which means that it selects only the elements that have a price greater than 10. The code fragment then applies a map operation on the filtered stream, which returns a new stream that contains the results of applying the given function to each element. The function is p -> p.getName (), which means that it returns the name of each element.

The code fragment then calls the collect method on the mapped stream, which performs a mutable reduction operation on the elements of the stream using a Collector. The Collector is Collectors.joining (“,”), which means that it concatenates the elements of the stream into a single String, separated by commas.

The code fragment then prints out the result of the reduction operation and the result of the collect operation, separated by a new line. The result of the reduction operation is 300.00, which is the sum of the prices of all ElectricProduct objects that have a price greater than 10. The result of the collect operation is CellPhone,ToyCar,Motor,Fan, which is the concatenation of the names of all ElectricProduct objects that have a price greater than 10.

Therefore, the output of the code fragment is:

300.00 CellPhone,ToyCar,Motor,Fan

References: Stream (Java SE 17 & JDK 17) - Oracle, Collectors (Java SE 17 & JDK 17) - Oracle

Question #:10

Given the code fragment:

```
Stream<String> s1 = Stream.of("A", "B", "C", "B");
Stream<String> s2 = Stream.of("A", "D", "E");
Stream.concat(s1, s2).parallel().distinct().forEach(element -> System.out.print(element));
```

What is the result:

A. ADEABCB // the order of element is unpredictable

B. ABCE

C. ABCDE // the order of elements is unpredictable

D. ABBCDE // the order of elements is unpredictable

Answer: D

Explanation

The answer is D because the code fragment uses the Stream API to create two streams, s1 and s2, and then concatenates them using the concat() method. The resulting stream is then processed in parallel using the parallel() method, and the distinct() method is used to remove duplicate elements. Finally, the forEach() method is used to print the elements of the resulting stream to the console. Since the order of elements in a parallel stream is unpredictable, the output could be any of the options given, but option D is the most likely.
References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Parallelizing Streams

Question #:11

Given:

```
interface IFace {
    public void m1();
    public default void m2() {
        System.out.println("m2");
    }
    public static void m3() {
        System.out.println("m3");
    }
    private void m4() {
        System.out.println("m4");
   }
}

class MyC implements IFace {
    public void m1() {
        System.out.println("Hello");
    }
}
```

Which two method invocation execute?

A. IFace myclassobj = new Myc ();

   myclassObj.m3 ();

B. Ifnce.m3 ();

C. iFace mucloassObj = new Myc ();

   myClassObj.m4();

D. new MyC() .m2 ();

E. IFace .,4():

F. IFace.m2();

Answer: D E

Explanation

The code given is an interface and a class that implements the interface. The interface has three methods, m1(), m2(), and m3(). The class has one method, m1(). The only two method invocations that will execute are D and E. D is a call to the m2() method in the class, and E is a call to the m3() method in the interface. References: https://education.oracle.com/products/trackp_OCPJSE17, 3, 4, 5

Question #:12

Given the code fragment:

```
Pet p = new Pet("Dog");
Pet p1 = p;
p1.name = "Cat";
p = p1;
System.out.println(p.name);
p = null;
System.out.println(p1.name);
```

What is the result?

A. Cat

Dog

B. A NullPointerException is thrown

Cat

Cat

C. Dog

Dog

D. Cat

null

Answer: D

Explanation

The answer is E because the code fragment creates a new Pet object with the name “Dog” and assigns it to the variable p. Then, it assigns p to p1. Next, it changes the name of p1 to “Cat”. Then, it assigns p1 to p. Finally, it sets p to null and prints the name of p and p1. The output will be “Cat” and “null” because p is set to null and p1 still points to the Pet object with the name “Cat”.

Question #:13

Given the course table:

COURSE_ID	COURSE_NAME	COURSE_FEE	COURSE_LEVEL
1021	Java Programmer	400.00	1
1022	Java Architect	600.00	2
1023	Java Master	600.00	2

Given the code fragment:

```
try (Connection con = DriverManager.getConnection(connectionString)) {
    Statement statement = con.createStatement(TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
    String qry = "UPDATE course SET course_fee = ? where COURSE_LEVEL = ?";
    PreparedStatement prStmt = con.prepareStatement(qry, TYPE_SCROLL_INSENSITIVE);
    prStmt.setDouble(1,600.00);
    prStmt.setInt(2,2);
    System.out.println(prStmt.executeUpdate());
}
catch(SQLException sqlException) {
    System.out.println(sqlException);
}
```


A. 2

B. false

C. true

D. 1

Answer: C

Explanation

The code fragment will execute the update statement and set the course fee of the course with ID 1021 to 5000. The executeUpdate method returns an int value that indicates the number of rows affected by the SQL statement. In this case, only one row will be updated, so the result variable will be 1. The if statement will check if the result is greater than 0, which is true, and print “Updated successfully”. Therefore, the output of the code fragment is true. References: https://education.oracle.com/products/trackp_OCPJSE17, https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487, https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Statement.html#executeUpdate(java.lang.Str

Question #:14

Given:

```
public class Test {
    public void sum(int a, int b) {
        System.out.print(" A");
    }
    public void sum(int a, float b) {
        System.out.print(" B");
    }
    public void sum(float a, float b) {
        System.out.print(" C");
    }
    public void sum(double... a) {
        System.out.print(" D");
    }
    public static void main(String[] args) {
        Test t = new Test();
        t.sum(10,15.25);
        t.sum(10, 24);
        t.sum(10.25,10.25);
    }
}
```


What is the result?

A. B A C

B. D A D

C. B A D

D. D D D

Answer: C

Explanation

The answer is C because the code demonstrates the concept of method overloading and type conversion in Java. Method overloading allows different methods to have the same name but different parameters. Type conversion allows values of one data type to be assigned to another data type, either automatically or explicitly. In the code, the class Test has four methods named sum, each with different parameter types: int, float, and double. The main method creates an instance of Test and calls the sum method with different arguments. The compiler will choose the most specific method that matches the arguments, based on the following rules:

- If there is an exact match between the argument types and the parameter types, that method is chosen.
- If there is no exact match, but there is a method with compatible parameter types, that method is chosen.

Compatible types are those that can be converted from one to another automatically, such as int to long or float to double.

- If there is more than one method with compatible parameter types, the most specific method is chosen. The most specific method is the one whose parameter types are closest to the argument types in terms of size or precision.

In the code, the following method calls are made:

- test.sum(10, 10.5) -> This matches the sum(int a, float b) method exactly, so it is chosen. The result is 20.5, which is converted to int and printed as 20 (B).
- test.sum(10) -> This does not match any method exactly, but it matches the sum(double a) method with compatible types, as int can be converted to double automatically. The result is 10.0, which is printed as 10 (A).
- test.sum(10.5, 10) -> This does not match any method exactly, but it matches two methods with compatible types: sum(float a, float b) and sum(double a, double b). The latter is more specific, as double is closer to the argument types than float. The result is 20.5, which is printed as 20 (D).

Therefore, the output is B A D. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Method Overloading in Java
- Type conversion in Java with Examples
- Java Method Overloading with automatic type conversions

Question #:15

Given:

```
1. class Item {
2.    String name;
3.    public static void display() {
4.       name = "Vase";
5.       System.out.println(name);
6.    }
7.    public void display(String design) {
8.       this.name += name;
9.       System.out.println(name);
10.   }
11. }
12. public class App {
13.   public static void main(String[] args) {
14.      Item i1 = new Item();
15.      i1.display("Flower");
16.   }
17. }
```


Which action enables the code to compile?

A. Replace 15 with item.display ("Flower");

B. Replace 2 with static string name;

C. Replace 7 with public void display (string design) {

D. Replace 3 with private static void display () {

Answer: C

Explanation

The answer is C because the code fragment contains a syntax error in line 7, where the method display is declared without any parameter type. This causes a compilation error, as Java requires the parameter type to be specified for each method parameter. To fix this error, the parameter type should be added before the parameter name, such as string design. This will enable the code to compile and run without any errors.
References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer

- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Java Methods

Question #:16

Given:

```
import java.io.Serializable;
public class Software implements Serializable {
  private String title;
  public Software(String title) {
    this.title = title;
    System.out.print("Software ");
  }
  public String toString() { return title; }
}

public class Game extends Software {
  private int players;
  public Game(String title, int players) {
    super(title);
    this.players = players;
    System.out.print("Game ");
  }
  public String toString() { return super.toString()+" "+players; }

}

import java.io.*;
public class AppStore {
  public static void main(String[] args) {
    Software s = new Game("Chess", 2);
    try(ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("game.ser"))) {
      out.writeObject(s);
    } catch (Exception e) {
      System.out.println("write error");
    }
    try(ObjectInputStream in = new ObjectInputStream(new FileInputStream("game.ser"))) {
      s = (Software)in.readObject();
    } catch (Exception e) {
      System.out.println("read error");
    }
    System.out.println(s);
  }
}
```


What is the result?

A. Software Game Chess 0

B. Software Game Software Game Chese 2

C. Software game write error

D. Software Game Software Game chess 0

E. Software Game Chess 2

F. Software Game read error

Answer: B

Explanation

The answer is B because the code uses the writeObject and readObject methods of the ObjectOutputStream and ObjectInputStream classes to serialize and deserialize the Game object. These methods use the default serialization mechanism, which writes and reads the state of the object's fields, including the inherited ones. Therefore, the title field of the Software class is also serialized and deserialized along with the players field of the Game class. The toString method of the Game class calls the toString method of the Software class using super.toString(), which returns the value of the title field. Hence, when the deserialized object is printed, it shows "Software Game Software Game Chess 2".
References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Serialization and Deserialization in Java with Example

Question #:17

Assume you have an automatic module from the module path display-ascii-0.2. jar. Which name is given to the automatic module based on the given JAR file?

A. Display.ascii

B. Display-ascii-0.2

C. Display-ascii

D. Display-ascii-0

Answer: C

Explanation

An automatic module name is derived from the name of the JAR file when it does not contain a module-info.class file. If the JAR file has an "Automatic-Module-Name" attribute in its main manifest, then

its value is the module name. Otherwise, the module name is derived from the JAR file's name by removing any version numbers and converting it to lower case. Therefore, for a JAR named display-ascii-0.2.jar, the automatic module name would be display-ascii, following these rules.

Question #:18

Given the code fragment:

```
int a = 2;
int b = ~a;
int c = a^b;
boolean d = a < b & a > c++;
System.out.println(d + " " + c);
boolean e = a > b && a > c++;
System.out.println(e + " " + c);
```

What is the result?

A. false 1

   false 2

B. true 1

   false 2

C. false 1

   ture 2

D. falase 0

   true 1

Answer: B

Explanation

The code fragment is comparing the values of a, b, and c using the < and > operators. The first comparison, d, is checking if a is less than b and greater than c. Since a is equal to 2, b is equal to -2, and c is equal to -4, this comparison will evaluate to true. The second comparison, e, is checking if a is greater than b and a is greater than c. Since a is equal to 2, b is equal to -2, and c is equal to -4, this comparison will evaluate to false. Therefore, the result will be true 1 false 2. References: Operators (The Java™ Tutorials > Learning the Java Language - Oracle

Question #:19

Given:

```
public class Main {
    void print(int i){
        System.out.println("hello");
    }
    void print(long j){
        System.out.println("there");
    }

    public static void main(String[] args) {
        new Main().print(0b1101_1010);
    }
}
```

A. Hello

B. Compilation fails

C. A NumberFormatException is thrown

D. there

Answer: B

Explanation

The code fragment will fail to compile because the parseInt method of the Integer class is a static method, which means that it can be invoked without creating an object of the class. However, the code is trying to invoke the parseInt method on an object of type Integer, which is not allowed. The correct way to invoke the parseInt method is by using the class name, such as Integer.parseInt (s). Therefore, the code fragment will produce a compilation error. References: Integer (Java SE 17 & JDK 17) - Oracle

Question #:20

Given the directory structure:

```
module1:
        p1\
           Doc.java
        p2\
           Util.java
```

Given the definition of the Doc class:

```
package p1;
    public sealed class Doc permits WordDoc {
}
```

Which two are valid definition of the wordDoc class?

A. Package p1;

   Public non-sealed class wordDoc extends Doc ()

B. Package p1;

   Public class wordDoc extends Doc ()

C. Package p1, p2;

   Public non-sealed class WordDoc extends Doc ()

D. Package p1, p2;

   Public sealed class WordDoc extends Doc ()

E. Package p1,

   non-sealed abstract class WordDoc extends Doc ()

F. Package p1;

   Public final class WordDoc extends Doc ()

Answer: A F

Explanation

The correct answer is A and F because the wordDoc class must be a non-sealed class or a final class to extend the sealed Doc class. Option B is incorrect because the wordDoc class must be non-sealed or final. Option C is incorrect because the wordDoc class cannot be in a different package than the Doc class. Option D is incorrect because the wordDoc class cannot be a sealed class. Option E is incorrect because the wordDoc class cannot be an abstract class. References: Oracle Certified Professional: Java SE 17 Developer, 3 Sealed Classes - Oracle Help Center

Question #:21

Given the code fragments:

```
class Test {
    volatile int x = 1;
    AtomicInteger xObj = new AtomicInteger(1);
}


and
public static void main(String[] args) {
    Test t = new Test();
    Runnable r1 = () -> {
        Thread trd = Thread.currentThread();
        while (t.x < 3 ) {
            System.out.print(trd.getName()+" : "+t.x+" : ");
            t.x++;
       }
    };
    Runnable r2 = () -> {
        Thread trd = Thread.currentThread();
        while (t.xObj.get() < 3) {
            System.out.print(trd.getName()+" : "+t.xObj.get()+" : ");
            t.xObj.getAndIncrement();
        }
    };
    Thread t1 = new Thread(r1,"t1");
    Thread t2 = new Thread(r2,"t2");
    t1.start();
    t2.start();
}
```


Which is true?

A. The program prints t1 : 1: t2 : 1: t1 : t2 : 2 : in random order.

B. The program prints t1 : 1 : t2: 1 : t1 : 2 : t2: 2:

C. The program prints t1 : 1: t2 : 1: t1 : 1 : t2 : 1 : indefinitely

D. The program prints an exception

Answer: B

Explanation

The code creates two threads, t1 and t2, and starts them. The threads will print their names and the value of the Atomic Integer object, x, which is initially set to 1. The threads will then increment the value of x and print

their names and the new value of x. Since the threads are started at the same time, the output will be in random order. However, the final output will always be t1 : 1 : t2: 1 : t1 : 2 : t2: 2: References: AtomicInteger (Java SE 17 & JDK 17) - Oracle

Question #:22

Which statement is true?

A. The tryLock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock.

B. The tryLock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.

C. The lock () method returns a boolean indicator immediately if it has managed to acquire the lock, otherwise it waits for the lock acquisition.

D. The Lock () method returns a boolean indicator immediately regardless if it has or has not managed to acquire the lock

Answer: A

Explanation

The tryLock () method of the Lock interface is a non-blocking attempt to acquire a lock. It returns true if the lock is available and acquired by the current thread, and false otherwise. It does not wait for the lock to be released by another thread. This is different from the lock () method, which blocks the current thread until the lock is acquired, and does not return any value. References: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/locks/Lock.html#tryLock(), 3, 4

Question #:23

Given the content of the in. tart file:

23456789

and the code fragment:

```
char[] buffer = new char[8];
int count = 0;
try(FileReader in = new FileReader("in.txt");
    FileWriter out = new FileWriter("out.txt")) {
        while((count = in.read(buffer)) != -1) {
            out.write(buffer);
        }
}
```

What is the content of the out .txt file?

A. 01234567801234

B. 012345678

C. 0123456789234567

D. 0123456789

E. 012345678901234

F. 01234567

Answer: D

Explanation

The answer is D because the code fragment reads the content of the in.txt file and writes it to the out.txt file. The content of the in.txt file is “23456789”. The code fragment uses a char array buffer of size 8 to read the content of the in.txt file. The while loop reads the content of the in.txt file and writes it to the out.txt file until the end of the file is reached. Therefore, the content of the out.txt file will be “0123456789”.

Question #:24

Given the code fragment:

```
Duration duration = Duration.ofMillis(5000);
System.out.print(duration);
duration = Duration.ofSeconds(60);
System.out.print(duration);
Period period = Period.ofDays(6);
System.out.print(period);
```

What is the result?

A. $SIM6D

B. PT5000PT60MP6D

C. PT5SPTIMP6D

D. 5000$60M6D

Answer: B

Explanation

The code fragment is creating a Duration object with a value of 5000 milliseconds, then printing it. Then, it is

creating another Duration object with a value of 60 seconds, then printing it. Finally, it is creating a Period object with a value of 6 days, then printing it. The output will be “PT5000PT60MP6D”. References: https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html, https://docs.oracle.com/javase/8/docs/api/java/time/Period.html

Question #:25

Given:

```
package com.transport.vehicle.cars;

public interface Car {
     int getSpeed();
}
```

and

```
package com.transport.vehicle.cars.impl;

import com.transport.vehicle.cars.Car;

public class CarImpl implements Car {
   private int speed;

   public CarImpl() {
      this(10);
   }

   public CarImpl (int speed) {
      this.speed = speed;
   }

    @Override
    public int getSpeed() {
       return speed;
   }
```


Which two should the module-info file include for it to represent the service provider interface?

A. Requires cm.transport.vehicle,cars:

B. Provides.com.transport.vehicle.cars.Car with com.transport.vehicle.cars. impt, CatImpI;

C. Requires cm.transport.vehicle,cars:

D. Provides.com.transport.vehicle.cars.Car impl,CarImp1 to com.transport.vehicle.cars. Cars

E. exports com.transport.vehicle.cars.Car;

F. Exports com.transport.vehicle.cars;

G. Exports com.transport.vehicle;

Answer: B E

Explanation

The answer is B and E because the module-info file should include a provides directive and an exports directive to represent the service provider interface. The provides directive declares that the module provides an implementation of a service interface, which is com.transport.vehicle.cars.Car in this case. The with clause specifies the fully qualified name of the service provider class, which is com.transport.vehicle.cars.impl.CarImpl in this case. The exports directive declares that the module exports a package, which is com.transport.vehicle.cars in this case, to make it available to other modules. The package contains the service interface that other modules can use.

Option A is incorrect because requires is not the correct keyword to declare a service provider interface. Requires declares that the module depends on another module, which is not the case here.

Option C is incorrect because it has a typo in the module name. It should be com.transport.vehicle.cars, not cm.transport.vehicle.cars.

Option D is incorrect because it has a typo in the keyword provides. It should be provides, not Provides. It also has a typo in the service interface name. It should be com.transport.vehicle.cars.Car, not com.transport.vehicle.cars.Car impl. It also has an unnecessary to clause, which is used to limit the accessibility of an exported package to specific modules.

Option F is incorrect because it exports the wrong package. It should export com.transport.vehicle.cars, not com.transport.vehicle.cars.impl. The impl package contains the service provider class, which should not be exposed to other modules.

Option G is incorrect because it exports the wrong package. It should export com.transport.vehicle.cars, not com.transport.vehicle. The vehicle package does not contain the service interface or the service provider class.
References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Java Modules - Service Interface Module - GeeksforGeeks
- Java Service Provider Interface | Baeldung

Question #:26

Which two code fragments compile?

A)

```
class L6 {
    public static void main(String[] args) {
        var x = new ArrayList<>();
        x.add(10);
        x.add("30");
        System.out.println(x);
    }
}
```

B)

```
class L2 {
    public void m(int x) {
        var x = 10;
    }
}
```

C)

```
class A {}
class B extends A{}
class L4 {
    public static void main(String[] args) {
        var x = new A();
        x = new B();
    }
}
```

D)

```
class L3 {
    public static void main(String[] args) {
        var a = 10;
        a = "30";
    }
}
```

E)

```
class L5 {
    public void m() {
        var strVar = null;
    }
}
```

A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

Answer: B E

Explanation

The two code fragments that compile are B and E. These are the only ones that use the correct syntax for declaring and initializing a var variable. The var keyword is a reserved type name that allows the compiler to infer the type of the variable based on the initializer expression. However, the var variable must have an initializer, and the initializer must not be null or a lambda expression. Therefore, option A is invalid because it does not have an initializer, option C is invalid because it has a null initializer, and option D is invalid because it has a lambda expression as an initializer. Option B is valid because it has a String initializer, and option E is valid because it has an int initializer.

https://docs.oracle.com/en/java/javase/17/language/local-variable-type-inference.html

Question #:27

Given the product class:

```
import java.io.*;
public class Product implements Serializable {
  private static float averagePrice = 2.99f;
  private String description;
  private transient float price;
  public Product(String description, float price) {
    this.description = description;
    this.price = price;
  }
  public void readObject(ObjectInputStream in)
        throws IOException, ClassNotFoundException {
    in.defaultReadObject();
    price = averagePrice;
  }
  public String toString() {
    return description+" "+price+" "+averagePrice;
  }
}
```

And the shop class:

```
import java.io.*;
public class Shop {
  public static void main(String[] args) {
    Product p = new Product("Cookie", 3.99f);
    try {
      try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("p.ser"))) {
        out.writeObject(p);
      }
      try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("p.ser"))) {
        p = (Product)in.readObject();
      }
    } catch (Exception e) { e.printStackTrace(); }
    System.out.println(p);
  }
}
```

What is the result?

A. Cookie 2.99 2.99

B. Cookie 3.99 2.99

C. Cookie 0.0 0.0

D. An exception is produced at runtime

E. Compilation fails

F. Cookie 0.0 2.99

Answer: E

Explanation

The code fragment will fail to compile because the readObject method in the Product class is missing the @Override annotation. The readObject method is a special method that is used to customize the deserialization process of an object. It must be declared as private, have no return type, and take a single parameter of type ObjectInputStream. It must also be annotated with @Override to indicate that it overrides the default behavior of the ObjectInputStream class. Without the @Override annotation, the compiler will treat the readObject method as a normal method and not as a deserialization hook. Therefore, the code fragment will produce a compilation error. References: Object Serialization - Oracle, [ObjectInputStream (Java SE 17 & JDK 17) - Oracle]

Question #:28

Given the code fragment:

```
List lst = new ArrayList();
lst.add("e1");
lst.add("e3");
lst.add("e2");

int x1 = Collections.binarySearch(lst, "e3");
System.out.println(x1);
Collections.sort(lst);
int x2 = Collections.binarySearch(lst, "e3");
System.out.println(x2);

Collections.reverse(lst);
int x3 = Collections.binarySearch(lst, "e3");
System.out.println(x3);
```

What is the result?

A. 0

B. 2

C. -2

D. 2

2

0

E. 1

1

1

F. 1

2

-4

Answer: B

Explanation

The code fragment uses the Collections.binarySearch method to search for the string “e3” in the list. The first search returns the index of the element, which is 2. The second search returns the index of the element, which is 0. The third search returns the index of the element, which is -4. The final result is
2. References: Collections (Java SE 17 & JDK 17) - Oracle

Question #:29

Given:

```
class A {public void mA() {System.out.println("mA");}}
class B extends A {public void mB() {System.out.println("mB");}}
class C extends B {public void mC() {System.out.println("mC");}}

public class App {
    public static void main(String[] args) {
        A bobj = new B();
        A cobj = new C();
        if (cobj instanceof B v) {
            v.mB();
            if (v instanceof C v1) { v1.mC(); }
        } else {
            cobj.mA();
        }
    }
}
```

What is the result?

A. Mb

MC

B. Mb

C. Mb

D. MA

E. mA

Answer: E

Explanation

The code snippet is an example of Java SE 17 code. The code is checking if the object is an instance of class C and if it is, it will print “mC”. If it is not an instance of class C, it will print “mA”. In this case, the object is not an instance of class C, so the output will be “mA”. References: Pattern Matching for instanceof - Oracle Help Center

Question #:30

Given:

Captions.properties file:

```
user = UserName
```

Captions_en.properties file:

```
user = User name (EN)
```

Captions_US.properties file:

```
message = User name (US)
```

Captions_en_US.properties file:

```
message = User name (EN - US)
```

and the code fragment:

```
Locale.setDefault(Locale.US);
Locale currentLocale = new Locale.Builder().setLanguage("en").build();

ResourceBundle captions = ResourceBundle.getBundle("Captions.properties", currentLocale);
System.out.println(captions.getString("user"));
```


What is the result?

A. User name (US)

B. The program throws a MissingResourceException.

C. User name (EN – US)

D. UserName

E. User name (EN)

Answer: B

Explanation

The answer is B because the code fragment contains a logical error that causes a MissingResourceException at runtime. The code fragment tries to load a resource bundle with the base name “Captions.properties” and the locale “en_US”. However, there is no such resource bundle available in the classpath. The available resource bundles are:

- Captions.properties
- Captions_en.properties

- Captions_US.properties
- Captions_en_US.properties

The ResourceBundle class follows a fallback mechanism to find the best matching resource bundle for a given locale. It first tries to find the resource bundle with the exact locale, then it tries to find the resource bundle with the same language and script, then it tries to find the resource bundle with the same language, and finally it tries to find the default resource bundle with no locale. If none of these resource bundles are found, it throws a MissingResourceException.

In this case, the code fragment is looking for a resource bundle with the base name “Captions.properties” and the locale “en_US”. The ResourceBundle class will try to find the following resource bundles in order:

- Captions.properties_en_US
- Captions.properties_en
- Captions.properties

However, none of these resource bundles exist in the classpath. Therefore, the ResourceBundle class will throw a MissingResourceException.

To fix this error, the code fragment should use the correct base name of the resource bundle family, which is “Captions” without the “.properties” extension. For example:

ResourceBundle captions = ResourceBundle.getBundle(“Captions”, currentLocale);

This will load the appropriate resource bundle for the current locale, which is “Captions_en_US.properties” in this case. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- ResourceBundle (Java Platform SE 8)
- About the ResourceBundle Class (The Java™ Tutorials > Internationalization)

Question #:31

Given the code fragment:

```
// Login time:2021-01-12T21:58:18.817Z
Instant loginTime = Instant.now();
Thread.sleep(1000);

// Logout time:2021-01-12T21:58:19.880Z
Instant logoutTime = Instant.now();

loginTime = loginTime.truncatedTo(ChronoUnit.MINUTES);   // line n1
logoutTime = logoutTime.truncatedTo(ChronoUnit.MINUTES);

if (logoutTime.isAfter(loginTime))
    System.out.println("Logged out at: " + logoutTime);
else
    System.out.println("Can't logout");
```

What is the result?

A. Logged out at: 2021-0112T21:58:19.880z

B. Logged out at: 2021-01-12T21:58:00z

C. A compilation error occurs at Line n1.

D. Can’t logout

Answer: B

Explanation

The code fragment is using the Java SE 17 API to get the current time and then truncating it to minutes. The result will be the current time truncated to minutes, which is why option B is correct. **References**:

- https://education.oracle.com/products/trackp_OCPJSE17
- https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487
- https://docs.oracle.com/javase/17/docs/api/java.base/java/time/Instant.html#truncatedTo(java.time.tempor

Question #:32

Given:

```
public enum Desig {
    CEO('A'), CMO('B'), CTO('C'), CFO('D');
    char c;
    private Desig(char c) {
        this.c = c;
    }
}
```

and the code fragment:

```
Arrays.stream(Desig.values()).dropWhile(s -> s.equals(Desig.CMO));
switch (Desig.valueOf("CMO")) {
    case CEO -> System.out.println("Executive");
    case CMO -> System.out.println("Marketing");
    case CFO -> System.out.println("Finance");
    case CTO -> System.out.println("Technical");
    default -> System.out.println("UnDefined");
}
```


What is the result

A. Marketing

   Finance

   Technical

B. Marketing

   Undefined

C. UnDefined

D. Marketing

Answer: C

Explanation

The code fragment is using the switch statement with the new Java 17 syntax. The switch statement checks the value of the variable desig and executes the corresponding case statement. In this case, the value of desig is “CTO”, which does not match any of the case labels. Therefore, the default case statement is executed, which prints “Undefined”. The other case statements are not executed, because there is no fall through in the new syntax. Therefore, the output of the code fragment is:

Undefined

Question #:33

Given the code fragments:

```
class Car implements Serializable {
    private static long serialVersionUID = 454L;
    String name;
    public Car(String name) { this.name = name; }
}

class LuxuryCar extends Car {        // line n1
    int flag_HHC;
    public LuxuryCar(String name, int flag_HHC) {
        super(name);
        this.flag_HHC = flag_HHC;
    }
    public String toString() {
        return name + " : " + flag_HHC;
    }
}
```

and:

```
public static void main(String[] args) {    // line n2
    Car b = new LuxuryCar("Wagon", 200);
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("car.ser"));
         ObjectInputStream ois = new ObjectInputStream(new FileInputStream("car.ser"));) {
            oos.writeObject(b);
            System.out.println((Car)(ois.readObject());          // line n3
    }
}
```


Which action prints Wagon : 200?

A. At line n1, implement the java.io, Serializable interface.

B. At line n3, replace readObject (0 with readLine().

C. At Line n3, replace Car with LuxurayCar.

D. At Line n1, implement the java.io.AutoCloseable interface

E. At line n2, in the main method signature, add throws IOException, ClassCastException.

F. At line n2, in the main method signature, add throws IoException, ClassNotFoundException.

Answer: F

Explanation

The code fragment is trying to read an object from a file using the ObjectInputStream class. This class throws an IOException and a ClassNotFoundException. To handle these exceptions, the main method signature should declare that it throws these exceptions. Otherwise, the code will not compile. If the main method throws these exceptions, the code will print Wagon : 200, which is the result of calling the toString method of the LuxuryCar object that was written to the file. References: ObjectInputStream (Java SE 17 & JDK 17) - Oracle, ObjectOutputStream (Java SE 17 & JDK 17) - Oracle

Question #:34

Given the code fragment:

```
record Product(int pNumber, String pName) {
    int regNo = 100;
    public int getRegNumber() {
        return regNo;
    }
}

public class App {
    public static void main(String[] args) {
        Product p1 = new Product (1111, "Ink Bottle");
    }
}
```


Which action enables the code to compile?

A. Replace record with void.

B. Remove the regNO initialization statement.

C. Make the regNo variable static.

D. Replace thye regNo variable static

E. Make the regNo variable public

Answer: E

Explanation

The code will compile if the regNo variable is made public. This is because the regNo variable is being

accessed in the main method of the App class, which is outside the scope of the Product class. Making the regNo variable public will allow it to be accessed from outside the class. References: https://education.oracle.com/products/trackp_OCPJSE17, https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487, https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html

Question #:35

Given the code fragment:

```
String myStr = "Hello Java 17";
String myTextBlk1 = """
                    Hello Java 17""";
String myTextBlk2 = """
                    Hello Java 17
                    """;
System.out.print(myStr.equals(myTextBlk1)+":");
System.out.print(myStr.equals(myTextBlk2)+":");
System.out.print(myTextBlk1.equals(myTextBlk2)+":");
System.out.println(myTextBlk1.intern() == myTextBlk2.intern());
```

A. True:false:true:true

B. True:true:false:false

C. True:false:true:false

D. True:false:false:false

Answer: C

Explanation

The code fragment compares four pairs of strings using the equals() and intern() methods. The equals() method compares the content of two strings, while the intern() method returns a canonical representation of a string, which means that it returns a reference to an existing string with the same content in the string pool. The string pool is a memory area where strings are stored and reused to save space and improve performance. The results of the comparisons are as follows:

- s1.equals(s2): This returns true because both s1 and s2 have the same content, “Hello Java 17”.
- s1 == s2: This returns false because s1 and s2 are different objects with different references, even though they have the same content. The == operator compares the references of two objects, not their content.
- s1.intern() == s2.intern(): This returns true because both s1.intern() and s2.intern() return a reference to the same string object in the string pool, which has the content “Hello Java 17”. The intern() method ensures that there is only one copy of each distinct string value in the string pool.

- “Hello Java 17” == s2: This returns false because “Hello Java 17” is a string literal, which is automatically interned and stored in the string pool, while s2 is a string object created with the new operator, which is not interned by default and stored in the heap. Therefore, they have different references and are not equal using the == operator.

References: String (Java SE 17 & JDK 17) - Oracle

Question #:36

Given the code fragment:

```
// line n1
String input = console.readLine("Input a number: ");
int number = Integer.parseInt(input);

if (number % 2 == 0) {
    System.out.println(number + " is even.");
} else {
    System.out.println(number + " is odd");
}
```

Which code line n1, obtains the java.io.Console object?

A)

```
Console console = Syst    em.cons    ole(Syst    em    .in);
```

B)

```
Console console = Cons    ole.get    Instance(    );
```

C)

```
Console console = Syst    em.cons    ole();
```

D)

```
Console console = new    Console(System.    in);
```

E)

```
Console console = new Console(new InputStreamReader(System.in));
```

A. Option A

B. Option B

C. Option C

D. Option D

E. Option E

Answer: A

Explanation

The code fragment is trying to obtain the java.io.Console object, which is a class that provides methods to access the character-based console device, if any, associated with the current Java virtual machine. The correct way to obtain the Console object is to call the static method Console console() in the java.lang.System class. This method returns the unique Console object associated with the current Java virtual machine, if any. Therefore, option A is correct, as it calls System.console() and assigns it to a Console variable. **References**:

- https://docs.oracle.com/javase/17/docs/api/java.base/java/io/Console.html
- https://docs.oracle.com/javase/17/docs/api/java.base/java/lang/System.html#console()
- https://education.oracle.com/products/trackp_OCPJSE17
- https://mylearn.oracle.com/ou/learning-path/java-se-17-developer/99487

Question #:37

Given the code fragment:

```
List<String> specialDays = List.of("NewYear","Valentines","Spring","Labour");
System.out.print(specialDays.stream().allMatch(s ->s.equals("Labour")));
System.out.print(" " + specialDays.stream().anyMatch(s ->s.equals("Labour")));
System.out.print(" " + specialDays.stream().noneMatch(s -> s.equals("Halloween")));
System.out.print(" " +specialDays.stream().findFirst());
```

What is the result?

A. False true true optional (Newyear)

B. 0110

C. True true false NewYear

D. 010 optional (Newyear)

Answer: A

Explanation

The code fragment is using the stream methods allMatch, anyMatch, noneMatch, and findFirst on a list of strings called specialDays. These methods are used to perform matching operations on the elements of a stream, such as checking if all, any, or none of the elements satisfy a given predicate, or finding the first element that matches a predicate1. The predicate in this case is that the string equals “Labour” or “Halloween”. The output will be:

- False: because not all of the elements in specialDays are equal to “Labour” or “Halloween”.
- true: because at least one of the elements in specialDays is equal to “Labour” or “Halloween”.
- true: because none of the elements in specialDays are equal to both “Labour” and “Halloween”.
- Optional[NewYear]: because the first element in specialDays that matches the predicate is “NewYear”, and the findFirst method returns an Optional object that may or may not contain a non-null value2.

References: Stream (Java SE 17 & JDK 17), Optional (Java SE 17 & JDK 17)

Question #:38

Given the code fragment:

```
Integer rank = 4;
switch (rank) {
    case 1,4 -> System.out.println("Range1");
    case 5,8 -> System.out.println("Range2");
    case 9,10 -> System.out.println("Range3");
    default -> System.out.println("Not a valid rank.");
}
```

What is the result?

A. Range 1

Range 2

Range 3

B. Range1

Note a valid rank.

C. Range 1

Range 2

Range 3

Range 1

Not a valida rank

D. Range 1

Answer: C

Explanation

value of the variable rank and executes the corresponding case statement. In this case, the value of rank is 4, so the first case statement is executed, printing “Range1”. The second and third case statements are also executed, printing “Range2” and “Range3”. The default case statement is also executed, printing “Not a valid rank”. References: Java Language Changes - Oracle Help Center

Question #:39

Given:

```
public class Test {
   public String attach1(List<String> data) {
       return data.parallelStream().reduce("w", (n,m) -> n+m, String::concat);
   }
   public String attach2(List<String> data) {
      return data.parallelStream().reduce((l, p)-> l+p).get();
   }

   public static void main(String[] args) {
      Test t = new Test();
      var list = List.of("Table", "Chair");
      String x= t.attach1(list);
      String y= t.attach2(list);
     System.out.print(x+ " "+y);
    }
}
```

What is the result?

A. Tablechair Tablechair

B. Wtablechair tableChair

C. A RuntimeException is thrown

D. wTableChair TableChair

E. Compilation fails

Answer: E

Explanation

The code fragment will fail to compile because the class name and the constructor name do not match. The class name is Furniture, but the constructor name is Wtable. This will cause a syntax error. The correct way to define a constructor is to use the same name as the class name. Therefore, the code fragment should change the constructor name to Furniture or change the class name to Wtable.

Question #:40

Given:

```
public class Weather {
     public enum Forecast {
          SUNNY, CLOUDY, RAINY;
          @Override
          public String toString() { return "SNOWY";}
     }

     public static void main(String[] args) {
          System.out.print(Forecast.SUNNY.ordinal() + " ");
          System.out.print(Forecast.valueOf("cloudy".toUpperCase()));
     }
}
```

What is the result?

A. 1 RAINY

B. Compilation fails

C. 1 Snowy

D. 0 CLOUDY

E. 0 Snowy

Answer: E

Explanation

The code is defining an enum class called Forecast with three values: SUNNY, CLOUDY, and RAINY. The toString() method is overridden to always return “SNOWY”. In the main method, the ordinal value of SUNNY is printed, which is 0, followed by the value of CLOUDY converted to uppercase, which is “CLOUDY”. However, since the toString() method of Forecast returns “SNOWY” regardless of the actual value, the output will be “0 SNOWY”. References: Enum (Java SE 17 & JDK 17), Enum.EnumDesc (Java SE 17 & JDK 17)

Question #:41

Given:

```
public class App {
    public int x = 100;

    public static void main(String[] args) {
        int x = 1000;
        App t = new App();
        t.myMethod(x);
        System.out.println(x);
    }
    public void myMethod(int x) {
        x++;
        System.out.println(x);
        System.out.println(this.x);
    }
}
```


What is the result?

A. 1001

1001

1000

B. 101

101

1000

C. 100

100

1000

D. 1001

100

1000

Answer: D

Explanation

The code fragment is using the bitwise operators & (AND), | (OR), and ^ (XOR) to perform operations on the binary representations of the integer values. The & operator returns a 1 in each bit position where both operands have a 1, the | operator returns a 1 in each bit position where either operand has a 1, and the ^ operator returns a 1 in each bit position where only one operand has a 1. The binary representations of the integer values are as follows:

- 1000 = 1111101000
- 100 = 1100100
- 101 = 1100101

The code fragment performs the following operations:

- x = x ^ y; // x becomes 1111010101, which is 1001 in decimal
- y = x ^ y; // y becomes 1100100, which is 100 in decimal
- x = x ^ y; // x becomes 1100101, which is 101 in decimal

The code fragment then prints out the values of x, y, and z, which are 1001, 100, and 1000 respectively. Therefore, option D is correct.

Question #:42

Given the code fragment:

```
ExecutorService executorService = Executors.newSingleThreadExecutor();
Set<Callable<String>> workers = new HashSet<Callable<String>>();
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "1";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "2";
    }
});
workers.add(new Callable<String>() {
    public String call() throws Exception {
        return "3";
    }
});
```

Which code fragment invokes all callable objects in the workers set?

A)

```
List<Future<String>> futures = executorService.invokeAny(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

B)

```
executorService.submit(cThreads);
```

C)

```
List<Future<String>> futures = executorService.invokeAll(workers);
for(Future<String> future : futures){
    System.out.println(future.get());
}
```

D)

```
for (int i=0; i<3;i++) {
    String result = executorService.invokeAny(cThreads);
    System.out.println(result);
}
```

A. Option A

B. Option B

C. Option C

D. Option D

Answer: C

Explanation

The code fragment in Option C invokes all callable objects in the workers set by using the ExecutorService's invokeAll() method. This method takes a collection of Callable objects and returns a list of Future objects representing the results of the tasks. The other options are incorrect because they either use the wrong method (invokeAny() or submit()) or have syntax errors (missing parentheses or semicolons). References: AbstractExecutorService (Java SE 17 & JDK 17) - Oracle

Question #:43

Which statement is true about modules?

A. Automatic and unnamed modules are on the module path.

B. Only unnamed modules are on the module path.

C. Automatic and named modules are on the module path.

D. Only named modules are on the module path.

E. Only automatic modules are on the module path.

Answer: C

Explanation

A module path is a sequence of directories that contain modules or JAR files. A named module is a module that has a name and a module descriptor (module-info.class) that declares its dependencies and exports. An automatic module is a module that does not have a module descriptor, but is derived from the name and contents of a JAR file. Both named and automatic modules can be placed on the module path, and they can be

resolved by the Java runtime. An unnamed module is a special module that contains all the classes that are not in any other module, such as those on the class path. An unnamed module is not on the module path, but it can read all other modules.

Question #:44

Given the code fragment:

```
String a = "Hello! Java";
System.out.print(a.indexOf("Java"));
a.replace("Hello!", "Welcome!");
System.out.print(a.indexOf("Java"));
StringBuilder b = new StringBuilder(a);
System.out.print(b.indexOf("Java"));
```

What is the result?

A. 81111

B. 8109

C. 777

D. 71010

E. 888

F. 7107

Answer: B

Explanation

The code fragment is creating a string variable “a” with the value “Hello! Java”. Then, it is printing the index of “Java” in “a”. Next, it is replacing “Hello!” with “Welcome!” in “a”. Then, it is printing the index of “Java” in “a”. Finally, it is creating a new StringBuilder object “b” with the value of “a” and printing the index of “Java” in “b”. The output will be 8109 because the index of “Java” in “a” is 8, the index of “Java” in “a” after replacing “Hello!” with “Welcome!” is 10, and the index of “Java” in “b” is 9. References: Oracle Java SE 17 Developer source and documents: [String (Java SE 17 & JDK 17)], [StringBuilder (Java SE 17 & JDK 17)]

Question #:45

Given the code fragment:

```
List<Integer> listOfNumbers = List.of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
```

Which code fragment returns different values?

A. int sum = listOfNumbers. parallelStream () reduce (5, Integer:: sum) ;

B. int sum = listOfNumbers. Stream () reduce (5, (a, b) -> a + b) ;

C. int sum = listOfNumbers. Stream () reduce (Integer:: sum) ; +5;

D. int sum = listOfNumbers. parallelStream () reduce ({m, n) -> m +n) orElse (5) +5;

E. int sum = listOfNumbers. Stream () reduce (0, Integer:: sum) + 5

Answer: C

Explanation

The answer is C because the code fragment uses a different syntax and logic for the reduce operation than the other options. The reduce method in option C takes a single parameter, which is a BinaryOperator that combines two elements of the stream into one. The method returns an Optional, which may or may not contain a value depending on whether the stream is empty or not. The code fragment then adds 5 to the result of the reduce method, regardless of whether it is present or not. This may cause an exception if the Optional is empty, or produce a different value than the other options if the Optional is not empty.

The other options use a different syntax and logic for the reduce operation. They all take two parameters, which are an identity value and a BinaryOperator that combines an element of the stream with an accumulator. The method returns the final accumulator value, which is equal to the identity value if the stream is empty, or the result of applying the BinaryOperator to all elements of the stream otherwise. The code fragments then add 5 to the result of the reduce method, which will always produce a valid value.

For example, suppose listOfNumbers contains [1, 2, 3]. Then, option A will perform the following steps:

- Initialize accumulator to identity value 5
- Apply BinaryOperator Integer::sum to accumulator and first element: 5 + 1 = 6
- Update accumulator to 6
- Apply BinaryOperator Integer::sum to accumulator and second element: 6 + 2 = 8
- Update accumulator to 8
- Apply BinaryOperator Integer::sum to accumulator and third element: 8 + 3 = 11
- Update accumulator to 11
- Return final accumulator value 11
- Add 5 to final accumulator value: 11 + 5 = 16

Option B will perform the same steps as option A, except using a lambda expression instead of a method reference for the BinaryOperator. Option D will perform the same steps as option A, except using

parallelStream instead of stream, which may change the order of applying the BinaryOperator but not the final result. Option E will perform the same steps as option A, except using identity value 0 instead of 5.

Option C, however, will perform the following steps:

- Apply BinaryOperator Integer::sum to first and second element: 1 + 2 = 3
- Apply BinaryOperator Integer::sum to previous result and third element: 3 + 3 = 6
- Return Optional containing final result value 6
- Add 5 to Optional value: Optional.of(6) + 5 = Optional.of(11)

As you can see, option C produces a different value than the other options, and also uses a different syntax and logic for the reduce operation. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Guide to Stream.reduce()

Question #:46

Given the code fragment:

```
abstract sealed interface SInt permits Story, Art {
default String getTitle() { return "Book Title" ; }
}
```

abstract sealed interface SInt permits Story, Art { default String getTitle() { return "Book Title" ; }

}

Which set of class definitions compiles?

A. Interace story extends STnt {}

   Interface Art extends SInt {}

B. Public interface story extends sInd {}

   Public interface Art extends SInt {}

C. Sealed interface Storty extends sInt {}

Non-sealed class Art implements Sint {}

D. Non-sealed interface story extends SInt {}

Class Art implements Sint {}

E. Non-sealed interface story extends SInt {}

Non-sealed interaface Art extends Sint {}

Answer: C

Explanation

The answer is C because the code fragment given is an abstract sealed interface SInt that permits Story and Art. The correct answer is option C, which is a sealed interface Story that extends SInt and a non-sealed class Art that implements SInt. This is because a sealed interface can only be extended by the classes or interfaces that it permits, and a non-sealed class can implement a sealed interface.

Option A is incorrect because interface is misspelled as interace, and Story and Art should be capitalized as they are the names of the permitted classes or interfaces.

Option B is incorrect because public is misspelled as public, and sInd should be SInt as it is the name of the sealed interface.

Option D is incorrect because a non-sealed interface cannot extend a sealed interface, as it would violate the restriction of permitted subtypes.

Option E is incorrect because both Story and Art cannot be non-sealed interfaces, as they would also violate the restriction of permitted subtypes.

References:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Sealed Classes and Interfaces in Java 15 | Baeldung
- Sealed Class in Java - Javatpoint

Question #:47

Which statement is true?

A. IllegalStateException is thrown if a thread in waiting state is moved back to runnable.

B. thread in waiting state consumes CPU cycles.

C. A thread in waiting state must handle InterrupedException.

D. After the timed wait expires, the waited thread moves to the terminated state.

Answer: C

Explanation

A thread in waiting state is waiting for another thread to perform a particular action, such as calling notify() or notifyAll() on a shared object, or terminating a joined thread. A thread in waiting state can be interrupted by another thread, which will cause the waiting thread to throw an InterruptedException and return to the runnable state. Therefore, a thread in waiting state must handle InterruptedException, either by catching it or declaring it in the throws clause. References: Thread.State (Java SE 17 & JDK 17), [Thread (Java SE 17 & JDK 17)]

Question #:48

Which statement is true about migration?

A. Every module is moved to the module path in a top-down migration.

B. Every module is moved to the module path in a bottom-up migration.

C. The required modules migrate before the modules that depend on them in a top-down ^migration.

D. Unnamed modules are automatic modules in a top-down migration.

Answer: B

Explanation

The answer is B because a bottom-up migration is a strategy for modularizing an existing application by moving its dependencies to the module path one by one, starting from the lowest-level libraries and ending with the application itself. This way, each module can declare its dependencies on other modules using the module-info.java file, and benefit from the features of the Java Platform Module System (JPMS), such as reliable configuration, strong encapsulation, and service loading.

Option A is incorrect because a top-down migration is a strategy for modularizing an existing application by moving it to the module path first, along with its dependencies as automatic modules. Automatic modules are non-modular JAR files that are treated as modules with some limitations, such as not having a module descriptor or a fixed name. A top-down migration allows the application to use the module path without requiring all of its dependencies to be modularized first.

Option C is incorrect because a top-down migration does not require any specific order of migrating modules, as long as the application is moved first and its dependencies are moved as automatic modules. A bottom-up migration, on the other hand, requires the required modules to migrate before the modules that depend on

them.

Option D is incorrect because unnamed modules are not automatic modules in any migration strategy. Unnamed modules are modules that do not have a name or a module descriptor, such as classes loaded from the class path or dynamically generated classes. Unnamed modules have unrestricted access to all other modules, but they cannot be accessed by named modules, except through reflection with reduced security checks. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- Migrating to Modules (How and When) - JavaDeploy
- Java 9 Modularity: Patterns and Practices for Developing Maintainable Applications

Question #:49

Given the code fragment:

```
class Book {
     String author;
     String title;
     Book(String authorName, String title) {
          this.author = authorName;
          this.title = title;
     }
}

class SortBook {
     public static void main(String[] args) {
         List books = List.of(new Book("A1","T1"), new Book("A2", "T2"), new Book("A1","T2")); // Line n1
         books.sort((Book a, Book b) -> a.title.compareTo(b.title));        // Line n2
         System.out.println(books);
     }
}
```

Which action sorts the book list?

A. At Line n2, replace books,sort() with books.stream().sort(0.

B. At line n1, convert books type to mutable ArrayList type.

C. At Line n1, convert type to mutable array type.

D. At Line n2, replace compareTo () with compare ().

Answer: D

Explanation

The code fragment is trying to sort a list of books using the Collections.sort() method. The correct answer is D, because the compareTo() method is not the correct way to compare two objects in a Comparator. The compare() method is the correct way to compare two objects in a Comparator and return an int value that indicates their order1. The compareTo() method is used to implement the Comparable interface, which defines the natural order of objects of a class2. The other options are incorrect because they either do not change the type of the list, which is already mutable, or they do not use the correct syntax for sorting a stream, which requires a terminal operation such as collect()3. References: Comparator (Java SE 17 & JDK 17), Comparable (Java SE 17 & JDK 17), Stream (Java SE 17 & JDK 17)

Question #:50

Daylight Saving Time (DST) is the practice of advancing clocks at the start of spring by one hour and

adjusting them backward by one hour in autumn.

Considering that in 2021, DST in Chicago (Illinois) ended on November 7th at 2 AM, and given the fragment:

```
ZoneId zoneID = ZoneId.of("America/Chicago");
ZonedDateTime zdt = ZonedDateTime.of(
    LocalDate.of(2021, 11, 7),
    LocalTime.of(1, 30),
    zoneID
);
ZonedDateTime anHourLater = zdt.plusHours(1);
System.out.println(zdt.getHour() == anHourLater.getHour());
```

What is the output?

A. true

false

B. False

false

C. true

true

D. false

true

Answer: A

Explanation

The answer is A because the code fragment uses the ZoneId and ZonedDateTime classes to create two date-time objects with the same local date-time but different zone offsets. The ZoneId class represents a time-zone ID, such as America/Chicago, and the ZonedDateTime class represents a date-time with a time-zone in the ISO-8601 calendar system. The code fragment creates two ZonedDateTime objects with the same local date-time of 2021-11-07T01:30, but different zone IDs of America/Chicago and UTC. The code fragment then compares the two objects using the equals and isEqual methods.

The equals method compares the state of two objects for equality. In this case, it compares the local date-time, zone offset, and zone ID of the two ZonedDateTime objects. Since the zone offsets and zone IDs are different, the equals method returns false.

The isEqual method compares the instant of two temporal objects for equality. In this case, it compares the instant of the two ZonedDateTime objects, which is derived from the local date-time and zone offset. Since DST in Chicago ended on November 7th at 2 AM in 2021, the local date-time of 2021-11-07T01:30 in America/Chicago corresponds to the same instant as 2021-11-07T06:30 in UTC. Therefore, the isEqual method returns true.

Hence, the output is true false. **References**:

- Oracle Certified Professional: Java SE 17 Developer
- Java SE 17 Developer
- OCP Oracle Certified Professional Java SE 17 Developer Study Guide
- ZoneId (Java Platform SE 8)
- ZonedDateTime (Java Platform SE 8)
- Time Zone & Clock Changes in Chicago, Illinois, USA
- Daylight Saving Time Changes 2023 in Chicago, USA