GROKKING THE JAVA SE 17 DEVELOPER CERTIFICATION

JAVINPAUL @JAVINPAUL

GROKKING THE JAVA SE 17 DEVELOPER CERTIFICATION

JAVINPAUL @JAVINPAUL

GROKKING THE JAVA SE 17 DEVELOPER CERTIFICATION

Preface

Welcome to the "Grokking the Java SE 17 Developer Certification." If you are preparing for Java SE 17 Developer certification with code 1Z0-829 then this book is for you.

In this book you will find a lot of practice questions on various topics related to 1Z0-829 certification from Oracle to become a certified Java developer

We understand that Java certification can be challenging, and preparing for them can be overwhelming. Our aim with this book is to help you grok the topics in-depth and ace your interviews with confidence.

In this book, I have provided not just practice questions but answers and explanations as well as links to learn more.

We hope that this book will be a valuable resource in your Java interview preparation journey. Let's get started and grok the Java interview together!

Table of Content

Arrays and Collections

1. Select statements that are true about arrays and collections in java.

1. ArrayList and Vector are thread safe.
2. Set can have duplicate elements.
3. Hashtable is thread safe
4. HashMap can have null as a key.
5. HashMap maintain the order of added items

Correct Answer: 3,4

Explanation:

1st is false because arraylist is not thread-safe, 2nd is false because set contains unique elements only, 3rd-hashtables are synchronized that's why thread safe, 4th- hashmap allows when key is null, 5th- hashmap does not keep order of added items.

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

2. Select statements that will compile.

```
var items<String> = new ArrayList<>();
```

1. var items = new ArrayList<Integer>();
2. List list = new ArrayList<Integer>();
   List list = new ArrayList<Integer>();
   list.add("item");
3. var items = new ArrayList<Integer>();
   items.add("item");
4. var items = new ArrayList<null>();

Correct Answer: 2,3,4

Explanation:

1st - we cannot set generic type for var, 2nd, 3rd, 4th options are allowed syntax in java, 5th generic type can not be null

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

3. Consider the code below: does it accept duplicates? If yes then how can we fix it? (Select only one statement)

```
public class Item {
   public String name;
   public int id;
   public Item(String name, int id) {
      this.name = name;
      this.id = id;
   }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        Item item = (Item) o;
        return false;
    }
    @Override
    public int hashCode() {
        return Objects.hash(1);
    }
}

¦in main method

        Set<Item> set = new HashSet<>();
        set.add(new Item("Item", 1));
        set.add(new Item("Item", 1));
```

1. Set it will not accept duplicates.
2. code will not compile.
3. equals must be fixed.
4. hashCode must be fixed.
5. Set uses its own equals and hashCode methods to identify duplicates.

Correct Answer: 3

Explanation:

3rd option is correct, equals method needs to be refactored. Instead of returning false there must be such code: return id == item.id && Objects.equals(name, item.name);

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

Concurrency

4. Select statements that are true about Concurrency in java?

1. threads can consist of multiple processes.
2. Order of execution of threads relies on thread priority and depends on the underlying platform(os, cores, etc).
3. Context switch is a mechanism of storing state of thread by scheduler that has already finished its execution.
4. Thread scheduler tries to minimize the number of context switch operations due to time cost.
5. threads can share memory.

Correct Answer: 2,4,5

Explanation:

1st is false because processes are formed from one or more threads. 2nd is true because threads can have priority and executed due to priority, by the way it also depends on number of cores, type of os and others factors, 3rd is false. Context switch - is mechanism to store state of executing thread that worked quantum of time and does not finished it's operation that's why its state is stored until next execution order, 4th is true. Scheduler of threads tries to minimize number of context switches because its takes time to store and then restore threads state. 5th is true because threads can share memory inside one process

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

5. Select statements that are true about Concurrency in java?

1. execution of threads in round-robin fashion may lead to starvation.
2. processes can have one thread only.
3. Threads can not share memory within a process.
4. Round-robin relies on a first come first serve algorithm.
5. multiple threads within the same process can share static fields value.

Correct Answer: 4,5

Explanation:

1st is false - round robin algorithm works with each thread for quantum of time. 2nd is false - processes can have more than one thread. 3rd is false - threads can share memory, 4th is true - RR alorithm relies on FIFO. 5th is true becasue static fields are shared among threads.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

6. Select statements that are true about Concurrency in java?

1. threads with same priority will be executed by last come first serve with round robin algorithm
2. JVM will wait until the end of the execution daemon thread.
3. User defined threads are daemon by default.
4. In a round-robin algorithm thread executed in quantum of time and if it is not finished context switch operation occurs.
5. When a user starts multiple threads he/she can predict order of execution.

Correct Answer: 4

Explanation:

1st is false - threads are executed as first come first serve when used with round robin, 2nd is false - JVM does not wait execution of daemon threads. 3rd is false - user defined threads are not daemon by default it needs to be set explicitly. 4th is true - in round robin algorithm threads executed in first come first serve order in a quantum of time. 5th is false - threads execution order is not predictable.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

7. What is the result of the code below?

```
var tmin = new Thread(() -> {
   System.out.print("T1");
});
var tmax = new Thread(() -> {
   System.out.print("T2");
});
tmin.run();
tmax.run();
System.out.println("main thread");
```

1. T2T1main thread
2. T1T2main thread
3. order is not predictable
4. main threadT1T2
5. none of the above

Correct Answer: 2

Explanation:

2nd option is true. When we call run() method directly then new thread is not get created, all code is executed normally in main thread sequentially.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Control Flow (Loop, switch, if/else)

8. Select valid syntax variant(s) of switch statements.

```
//First
int num = 1;
String label = switch (num) {
   case 1 -> "one";case 2 -> "two";
};
//Second
String label2 = switch (num) {
   case 2 -> "two"; case 3,4,5 -> "three, four, five";
   default -> throw new
IllegalStateException("Unexpected value: " + num);
};
//Third
var month = Month.APRIL;
String monthSeason = switch (month) {
   case APRIL -> "spring"; case JUNE, JULY ->
"summer";
};
```

1. First, Third

2. All are correct

3. First, Second

4. Second

5. Second, Third
6. None of them

Correct Answer: 4

Explanation:

First sample is incorrect - it needs default case, Second has all cases, Third option needs default case or just need to be filled with other months cases.

- ➢ https://docs.oracle.com/en/java/javase/17/language/switch-expressions.html

Date/Time API

9. Select statements that are true about Date/Time API in java.

1. It is thread safe.
2. Classes are located in java.datetime package.
3. LocalDateTime.now() can show date, time and zone id of current place.
4. The id of months starts from 0.
5. LocalDate methods can be chained because they do not return null

Correct Answer: 1,5

Explanation:

As documents says new Date/Time api is thread safe. 2nd is false - classes located in java.time package, 3rd is false - LocalDateTime.now() can show current date and time. 4th is false - months are started from 1, 5th is true - LocalDate methods can be chained because they do not return null

- ➢ https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

10. Select statements that will get date and time with timezone. (Example of timezone for asia it is +06.00)

1. LocalDateTime.now().atZone(ZoneId.systemDefault())

2. GMTDateTime.now()
3. ZonedDateTime.now()
4. OffsetDateTime.now()
5. LocalDate.getDefaultZoneId()

Correct Answer: 1, 3, 4

Explanation:

1th, 3rd, 4th variants return datetime with timezone, 2nd is false - no such class. 5th is false it will not return even datetime

- ➢ https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

Exception Handling

11. Select statements that are true about exceptions in java.

1. runtime exceptions must be enclosed in try catch statement
2. custom exception class that inherits from Exception class may or may not be enclosed with try catch block or added exception to method signature.
3. RuntimeException do not inherit the Exception class.
4. When error occurs runtime system searches for method that will handle this error
5. try with resource works with classes that implements AutoCloseable

Correct Answer: 4,5

Explanation:

Runtime exceptions do not need to be enclosed within try catch, Exceptions and its child classes must be enclosed with try catch or added exception to method signature, Runtime exception extends Exceptions class. Yes when error occurs runtime system search method that will handle it, yes try with resource works with classes that implement AutoCloseable

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

12. Select statements that are true about exceptions in java.

1. Error and runtime exceptions are known as unchecked exceptions.
2. java has 4 types of exceptions: exception, error, runtime exceptions and checked exceptions.
3. When a program is trying to access disk but throws external error due to malfunction of hardware is the subject of runtime exception.
4. when using try with resource then there is a need to close the resource explicitly.
5. Error and Exception are subclasses of Throwable.

Correct Answer: 1,5

Explanation:

1st is true - Error and runtime exceptions are unchecked. 2nd is false - java has 3 types : error, exception, runtime exception. 3rd is false - hardware exceptions are subjects of error. 4th is false try with recourses close resource implicitly. 5th is true they are child classes of Throwable

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

13. Select variants that will compile.

1. try {throw new IndexOutOfBoundsException();}
2. try {throw new IndexOutOfBoundsException();}finally {}
3. try {throw new IndexOutOfBoundsException();}catch {}
4. try {throw new IndexOutOfBoundsException();}catch(Exception e)
5. String url = "jdbc/postgresql/localhost:5432/mydatabase"; try (Connection conn = DriverManager.getConnection(url)) {}catch(Exception e){}

Correct Answer: 2,5

Explanation:

1st will not compile because it requires catch or finally. 2nd will compile - it has all needed parts. 3rd need to add type of exceptions ex: catch(Exception e) {}. 4th will not compile because it needs brackets {}. 5th is valid

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Garbage collection

14. Before the code below ends execution which of the following variants is true for the garbage collection mechanism?

```
String a,b,c = "c";
a = new String("a");
b = a;
a = null;
c = b;
```

1. new String("a") object will be eligible after "a=null" line
2. String "c" is eligible for garbage collection
3. String "c" is a string literal
4. new String("a") has multiple references that's why it will stay in a heap until the end
5. after the "a=null" line new String("a") object lost all its references
6. none of the above

Correct Answer: 3, 4

Explanation:

1st is false-String("a") has b as reference so then it is no eligible. 2nd is false - string "c" is string literal and not stored in heap so then it will not be collected by gc. 3rd is true "c" is a string literal. 4th is true String("a") has multiple references that's why it will stay in a heap until the end of program execution. 5th is false because in this case variable a is only who lost its reference to object String("a"). 6th is false

- https://blogs.oracle.com/javamagazine/post/understanding-garbage-collectors

Java OOA instanceof

15. Select statements that are correct alternatives of code below using pattern matching.

```
      if(num instanceof Integer) {
         Integer intNum = (Integer)num;
         if(intNum%2==0) {
System.out.println("Even:"+intNum); }
      }
```

1.
   ```
   if(num instanceof Integer intNum) {
            if(intNum%2==0) {
   System.out.println("Even:" + intNum);}
   }
   ```

2.
   ```
   if(num instanceof Integer intNum && intNum%2==0) {
            System.out.println("Even:" + intNum);
         }
   ```

3.
   ```
   if(num instanceof final Integer intNum) {
            if(intNum%2==0) {
   System.out.println("Even:" + intNum);}
         }
   ```

4.
   ```
   if(num instanceof final Integer intNum && intNum
   %2==0) {
            System.out.println("Even:" + intNum);
         }
   ```

5. none of the above

Correct Answer: 1,2,3,4

Explanation:

1,2,3,4 are all correct alternatives of code given. Due to new syntax instanceof we can check instance with its reference variable and use it further in a code. 5th is false

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-instanceof-operator.html

Java OOA Overloading, Overriding, Interfaces

16. Select statements that are true about Java Object Oriented Approach.

1. allow multiple inheritance (class has multiple parents).
2. java.lang.Object does not have a parent class.
3. The child class inherits methods from the parent class.
4. class can extend from a single class and implement a single class only.
5. child class must implement methods from interface
6. equals() method can not be overriden

Correct Answer: 2,3,5

Explanation:

1st is false class can have single parent only, 2nd is true - java.lang.Object is base class and highest in java classes hierarchy and it has no parent. 3rd is true because child class inherit parent's methods. 4th is false - it can extend from single class but implement multiple. 5th is true - interface methods must be implemented. 6th is false - we can override equals() method

- ➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

17. Select statements that are true about Java Object Oriented Approach.

1. java.lang.Object is the parent class for all classes.
2. primitive types extend java.lang.Object.
3. final class can not be inherited
4. record class can be inherited.
5. none of the above

Correct Answer: 1,3

Explanation:

1st is true - java.lang.Object is base class and all classes are derived from it directly and indirectly. 2nd is false - primitive types does not extend java.lang.Object. 3rd is true final class cannot be inherited. 4th is false because record class is final by default and can not be inherited. 5th is false

- ➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

18. Select statements that are true about Java Object Oriented Approach.

1. a single class cannot implement multiple interfaces.
2. java class name and name of file must be the same.

3. java files can have only one top-level class and multiple nested classes.

4. all classes must have at least one public top-level class.

5. sealed is the new class modifier

6. none of the above

Correct Answer: 2,3,5

Explanation:

1st is false - single class can implement multiple interfaces. 2nd is true - java class name and name of file must be the same. 3rd is true java files can have only one top-level class and can have multiple nested classes. 4th is false - class can have top-level class with default modifier. 5th is true - java added sealed keyword that can be inherited only for permitted classes. 6th is false

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

19. Select statements that are true about Java Object Oriented Approach.

1. keyword this - is used to reference instance variables.

2. constructor of class can be of type void.

3. constructors can not have return type.

4. final class and record class can not be inherited.
5. when instance and method variable name is the same then code will not compile
6. none of the above

Correct Answer: 1,3,4

Explanation:

1st is true - by using this keyword we can reference instance variables, it is useful when name of method variables and instance variables are the same. 2nd is false - constructors do not have return type. 3rd is true - see 2nd variant. 4th is true - final and record class are final classes that’s why cannot be inherited.5th is false - code will compile variable names will not interfere, if we need to call instance variable then use 'this' keyword. 6th is false.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

20. Select statements that are true about Java Object Oriented Approach.

1. keyword this - is used to reference instance variables.
2. a child class cannot have fields with the same name as its parent class.
3. keyword super - is used to reference parent fields.
4. all fields, even private ones, are accessible in child class using keyword super.

5. child class fields and methods can be the same as parent class fields and methods.

6. @Override is used to overload method

Correct Answer: 1,3,5

Explanation:

1st is true - keyword this is used to reference instance fields of a class. 2nd is false - child can have fields that are same with parent fields. 3rd is true - keyword super used to access parent fields. 4th is false - private fields are only accessible within declared class. 5th is true - child class can have same field and method names as it's parent class. 6th is false - @override is used in children classes to implement method from interface or parent class, it tells compiler that this method name and return type must be the same as parent and if not then it throws an exception.

- ➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

JDBC

21. Select all types of jdbc statements.

1. FutureStatement
2. CallableStatement
3. PreparedStatement
4. Statement
5. AsyncStatement
6. AsyncJDBCStatement

Correct Answer: 2,3,4

Explanation:

CallableStatement, Statement, PreparedStatement - are correct variants, others are fake names.

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

22. Select statements that will help to connect to the postgresql database.

1. Download needed database driver.
2. Open connection using DriverManager.getConnection("jdbc://localhost/mydb"‹).

3. Open connection using DriverManager.getConnection("jdbc:postgresql//localhost/mydb"‹)
4. set database driver in classpath when run application from command line.
5. Starting from java 8 version no need to download database driver because it is already contained in java as a separate module.
6. In order to use the java.sql package developer must require the module where the package is located.

Correct Answer: 1,3,4

Explanation:

1st is true - to connect to postgresql db we need to download corresponding driver, 2nd is false - we also need to write explicitly type of database in DriverManager.getConnection() see 3rd option. 3rd is true - we use correct path to our database, 4th is true - when java program compiled and executed from cli then we need to set database driver to classpath. 5th is false - java 8 does not contain drivers. 6th is false - java.sql package already exported in java.base module so we don't need to require it explicitly.

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Lambda and Functional Interface

23. Select statements that are true about lambda and functional interface.

1. Lambda used with a functional interface exactly with one abstract method.
2. Functional interfaces may have more than one abstract method.
3. Interface may or may not have @FunctionalInterface annotation.
4. Interface must be extended in order to be used with lambda.
5. Lambda cannot accept parameters.

Correct Answer: 1, 3

Explanation:

Functional interface used with lambda and can have only one abstract method, it can be declared with @FunctionalInterface annotation or without it. Lambda can have parameters.

- ➢ https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

24. Select statements that are true about lambda and functional interface.

1. () -> {}   - is valid lambda.
2. x -> { x.startsWith("a") } - is valid lambda.
3. Interface may or may not have @FunctionalInterface annotation.
4. Functional interface may have multiple static methods.
5. Functional interface can not have a default method.
6. Functional interface must have only one method.

Correct Answer: 1,3,4

Explanation:

1st is valid code. 2nd is false - need to set semicolon. 3rd is true - Any interface with one abstract method considered as @FunctionalInterface. 4th is true - interface can have multiple static methods. 5th is false - FunctionalInterface can have default method. 6th is false - FunctionalInterface can have one abstract, default and static methods.

- ➢ https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

Localization

25. Select statements that are true about Localization in java.

1. En_Us is valid Locale identifier.
2. language code must be lowercase.
3. language code and country must be separated by underscore
4. NumberFormat class is thread safe
5. FR_fr is a valid Locale identifier.
6. none of the above.

Correct Answer: 2,3

Explanation:

Locale code must be of language code in lower case(en, fr, ch) seperated with underscore and country code in uppercase(US, FR, CH), all other variants are invalid. NumberFormat is not thread safe.

- ➢ https://docs.oracle.com/javase/tutorial/i18n/

26. Select statements that are true about Localization in java.

1. Output of numbers may vary when parsed using the NumberFormat parse() method.

2. CompactNumberFormat uses LONG as default style type.
3. There are three types of Locale categories can be set: DISPLAY, FORMAT, COMPACT
4. CompactNumberFormat extends DecimalFormat
5. Category.DISPLAY is used to set format of number and currency
6. CompactNumberFormat helps to format numbers and show them in limited space.

Correct Answer: 1,6

Explanation:

1st is true - according to default Locale type output of parse method will vary. 2nd is false - default style type is Short. 3rd is false - 2 types only a) DISPLAY - used to represent the default locale for displaying user interfaces and FORMAT - used to represent the default locale for formatting dates, numbers, and/or currencies. 4th is false - CompactNumberFormat extends NumberFormat, 5th is false - DISPLAY - used to represent the default locale for displaying user interfaces. 6th is true - CompactNumberFormat used to show numbers in limited space areas.

- https://docs.oracle.com/javase/tutorial/i18n/

Math API

27. What will be the output of the code below?

```
      double d = 2.15;
      var round = Math.round(d);
      var ceil = Math.ceil(d);
      var floor = Math.floor(d);
      System.out.println("" + (round == ceil)+(round
== floor)+(ceil == floor));
```

1. falsefalsefalse
2. truetruefalse
3. falsetruetrue
4. falsetruefalse
5. truetruetrue
6. none of the above

Correct Answer: 4

Explanation:

4th is true - Math.round converts number to long value 2, Math.ceil will turn into 3.0 and Math.floor to 2.0. Compare 2 == 3.0 will be false, Compare 2==2.0 is true and compare 3.0 with 2.0 is false.

- ➢ https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Modules

28. Select statements that are true about java modules.

1. java modules now are mandatory to use
2. provide new level of security and access to packages and classes
3. public class is always accessible from every part of project when we use modules
4. public class may or may not be accessible from every part of project due to config in module-info.java
5. packages in module may or may not be reusable due to config in module-info.java
6. core java is also made from modules

Correct Answer: 2,4,5,6

Explanation:

1st is false - writing code as module is not mandatory. 2nd is true - modules gives use new level of security and access to code. 3rd is false - public class is not accesible from module that is not exported. 4th is true - public class is accessible if package is exported in module-info.java. 5th is true - package must be exported to be used by other modules. 6th is true - java now is consist of modules.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

29. Select statements that are true about java modules.

1. module can require another module and export one or more packages
2. exports somepackagename - used to export package
3. conceals somepackagename - used to hide package from outside world
4. in some java.core packages public class is accessible even in concealed package
5. module name may or may not be unique
6. module is the new level of encapsulation

Correct Answer: 1,2,6

Explanation:

1st and 2nd options are valid. 3th is false - no such keyword conceals. 4th is false concealed package classes are not accessible from outside. 5th is false - module name must be unique. 6th - module is new level of encapsulation.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

30. Consider such a scenario with java modules. Select correct statements.

```
//SayHello.java
package m.hello;
public class SayHello {
public void sayHello() {System.out.println("Hello
java 17");}}
//module-info.java
module hello {exports m.hello;}

//Main.java
package m.main;
import m.hello.SayHello;
public class Main {
public static void main(String[] args) {new
SayHello().sayHello();}}
//module-info.java
module main {requires hello;}
//modules located in mymodules folder;
```

1. code will not compile
2. module main requires method hello
3. hello module needs explicitly require module where class System is located
4. javac module-source-path   mymodules -m hello -d out; will compile hello module
5. code successfully outputs Hello java 17
6. java module-path out main/m.main.Main - command would run program

Correct Answer: 4, 5, 6

Explanation:

1st is false - code will compile. 2nd is false - there is no method hello. 3rd is false - System class located in java.base module. As we know java.base can be used without explicitly require it. 4th is true - it is way how we can compile module. 5th is true. 6th is true - it is the way how we run compiled java class in module.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

31. Select statements that are true about java modules.

1. Java EE modules are removed from core java
2. java.base is the core package in java;
3. package can be exported only for specific module
4. modules are divided into normal and open modules
5. requires transitive, requires static are used to import needed package
6. java.se is a module that combine all other java standard packages

Correct Answer: 1,3,4,6

Explanation:

1st is true - Java EE modules are removed from java 17. 2nd is false - java.base is a core module. 3rd is true - package can be exported to whole world or only specific module. 4th is true - modules are divided into normal and open modules by definition. 5th is false - requires used with module names not packages. 6th is true - java.se is a module that combine all java standard modules

- (https://docs.oracle.com/en/java/javase/11/docs/api/java.se/module-graph.png).
- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

Arithmetic/Boolean Expressions (Operators, Promotion, Casting)

32. What is the result of the following code snippet?

```
1       int a = 1;
2       int b = (int)20l;
3       char d = 10;
4       a += (++b) + (d++);
5       System.out.println(a);
```

1. will not compile because of cast operation on line 2
2. result will be 33
3. on line 4 add operation between different types does not allowed
4. Program will output 32
5. on line 3 assignment of integer to char d does not allowed
6. none of the above

Correct Answer: 4

Explanation:

we have 3 int variables: a = 1, b = 20, d= 10, operation a +=(++b) + (d++) we can rewrite into a = a + (++b) + (d++) then according to operator precedence evaluate (++b) left to right so value of variable b is 21, then evaluate (d++) left to right and value will return 10 and then incremented to 11, so at the moment of addition we will have a = 1 + 21 + 10. Result output is a = 32

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

33. What is the result of the following code snippet and type of numb ?

```
long num = 10;
num +=10;
int numa = 4;
var numb = num + numa;
float numc = 10;
var numd = numc + numb;
System.out.println(numd++);
```

1. code will not compile
2. type of numb is float; result of output 34.0
3. type of numb is int; result of output 34.0
4. type of numb is long; result of output 34
5. type of numb is int; result of output 34
6. type of numb is float; result of output 35.0

Correct Answer: 2

Explanation:

Code is valid. Variable num after line 2 will be 20l, after line 4 value of numb will be 24 of type long. After line 6 value of numd will be 34.0 of type float. By the rule sum of different types will result to bigger size type, so sum of float and long will be of type float. Sum of long and int will be long. At line 7 compiler will return numd (34.0) and then increment, so result will be 34.0 of type float.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

Record

34. Which of the following is true for a class that is created with the record keyword in java? Select all possible variants

1. record class is mutable; need to explicitly create hashCode(), equals() and toString() methods
2. record class is immutable, no need to create hashCode(), equals() and toString() methods; all fields are private and final
3. record is mutable; all fields are private and final
4. record class removes boilerplate code such as creation of methods hashCode(), equals() and toString()
5. record class must extend sealed class

Correct Answer: 2,4

Explanation:

Record class is immutable and must not extend sealed class, so variants 1,3,5 are invalid. Record is immutable its fields are private and final by default and no need to override Object methods because its done by compiler implicitly.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

35. Select statements that are true about record.

1. has private methods and private final fields
2. record class is simple data carrier
3. has public access methods and private final fields
4. new instance of record cannot be created using keyword new
5. record class can extend sealed class
6. record class is mutable

Correct Answer: 2, 3, 5

Explanation:

Record class is a simple data carrier that has public access methods and private final field(s). New instance of record is created using keyword 'new' as usual class and it is immutable.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

Sealed Classes

36. Select statements that are true about sealed classes.

1. subclasses are free to extend or not their parent; subclasses must be final, sealed or non-sealed class
2. all subclasses must extend their parent; it is optional for subclasses to be final, sealed or non-sealed class
3. all subclasses must extend their parent; subclasses must be final, sealed or non-sealed class
4. all subclasses must extend their parent; subclasses cannot have its own child class
5. subclasses are free to extend or not their parent; subclass must be final, sealed or non-sealed class; subclass cannot have its own child class

Correct Answer: 3

Explanation:

Any class which comes after permits keyword in sealed class must extend that sealed class. Child class must have final, sealed or non-sealed modifier. Child class can have its own children classes. So only 3rd option is valid.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;

- https://openjdk.org/jeps/409

37. Select statements that are true about sealed classes.

1. a sealed class can have restricted number of child classes
2. child classes may or may not extend parent class
3. child classes may or may not have modifiers final, sealed, non-sealed
4. before java added sealed feature developers used final keyword to restrict class to be inherited
5. child classes must have final, sealed or non-sealed modifier
6. sealed class feature came to replace final keyword

Correct Answer: 1,4,5

Explanation:

1st is true - sealed class restricts other classes to inherit itself. Only that classes that comes after keyword permits are allowed to extend sealed class. 2nd is false - child class must extend parent sealed class. 3rd is false - child of sealed class must have final, sealed or non-sealed modifier. 4th is true - before sealed class feature, developers set class as final to close it to be inherited. 5th is true - child of sealed class must have final, sealed, non-sealed modifier.6th is false -sealed class feature came to restrict inheritance mechanism.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- https://openjdk.org/jeps/410

Streams

38. Select statements that are true about streams in java.

1. intermediate and terminal operations return type is a stream
2. consist of source, intermediate and terminal operation
3. there can be multiple intermediate operations and only one terminal
4. streams forEach outer scope variables that are not started with Atomic* must be final
5. intermediate operations use lazy evaluations
6. none of the above

Correct Answer: 2,3,4,5

Explanation:

Streams - consist of source, intermediate and terminal operations. It can have multiple intermediate and single terminal operation. ForEach method's variables that were declared in outer scope must be final or Atomic*. Streams intermediate operations use lazy evaluations and executed when terminal operation is called.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

39. Select correct creation of streams from data: one, 1, two, 2, three

1. var stream = Stream.of("one", 1, "two", 2, "three");
2. var list = List.of("one", 1, "two", 2, "three");
3. List<Object> list = List.of("one", 1, "two", 2, "three"); list.stream();
4. var list = List.of("one", 1, "two", 2, "three"); list.parallelStream();
5. var stream = Stream.of("one", 1, "two", 2, "three", 3);
6. Stream<String> stream = Stream.of("one", "1", "two", "2", "three");

Correct Answer: 1,3,4,6

Explanation:

2nd is invalid - because it is list and not stream. 5th option is invalid because there is extra number 3. Others are valid.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

40. Select statements that will create a new stream with even numbers, distinct numbers and get maximum value from the list below. var intStream = List.of(0,1,2,3,3,4,5,6,7,7,8,9);

1. intStream.stream().filter(i -> i%2 == 0); intStream.stream().distinct(); intStream.stream().max()
2. intStream.stream().filter(i -> i%2 != 0); intStream.stream().distinct(); intStream.stream().max()
3. intStream.stream().filter(i -> i%2 == 0); intStream.stream().distinct(); intStream.stream().max(Integer::compare)
4. intStream.stream().filter(i -> i%2 == 0); intStream.stream().distinct();intStream.stream().sorted(Comparator.reverseOrder()).findFirst()
5. intStream.stream().filter(i -> !(i%2 != 0)); intStream.stream().distinct(); intStream.stream().max(Integer::compare)
6. none of the above

Correct Answer: 3,4,5

Explanation:

1st invalid - max() function needs java.util.Comparator as a parameter. 2nd invalid because it will return odd numbers. 3rd is valid - first stream filters numbers that are even, second stream gets distinct numbers, third stream gets maximum value, 4th is valid - first and second stream works as in 3rd variant, third stream orders items in descending order and returns first value. 5th is valid - first stream takes negation of odd numbers condition that’s why stream will return even numbers other streams are the same as 3rd variant, 6th invalid.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

41. Select the correct value of the num.

```
var numStream = List.of(0,1,22,3,-13, 4, -5, 6, 7,7,8,11);
var num = numStream
.stream()
.distinct().filter(i -> i % 3 ==
1).sorted().findFirst().get();
```

1. 2
2. 4
3. 1
4. 3
5. 0
6. code will not compile

Correct Answer: 3

Explanation:

Stream filters numbers that has remainder 1 when divided by 3, so new stream will be 1, 22, 4, 7 which after sorted() method call will be 1, 4, 7, 22 then first item will be 1.

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

42. Select the correct value of the number variable.

```
var mixNumStream =
List.of(Integer.valueOf(37),5,22,13, 47,-113);
var number = mixNumStream.stream()
.distinct()
.filter(i -> i % 5 ==
2).sorted(Comparator.reverseOrder()).findFirst().get();
```

1. 22
2. 13
3. 37
4. 47
5. -113
6. code will not compile

Correct Answer: 4

Explanation:

stream filters 37, 22,47 then will sort in descending order and returns first result which is 47.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

Text Block

43. Select correct variants of text block usage.

1.
```
String title = """HelloWorld
          """;
```

2.
```
String title = """
          HelloWorld""";
```

3.
```
var hello = "Hello";var world = "World";
    String title = """ """ + hello + " " + world;
```

4.
```
var hello = "Hello";var world = "World";
   String title = """
          ${} ${}
         """.setParam(hello).setParam(world);
```

5.
```
var hello = "Hello";  var world = "World";
String title = """
       ${hello} ${world}""";
```

Correct Answer: 2, 5

Explanation:

1st is invalid - text must begin on next line after opening triple quotes. 2nd is valid - text value on second line. 3th is invalid - open and close quotes must be on different rows. 4th is false - no such things as setParam 5th is true - it will print ${hello} ${world}

➢ https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

44. Select statements that are true about the text block and code below.

```
     String name = "Jane";
     var varName = """
          Jane
          """;
     System.out.print(name.equals(varName));

System.out.print((name+"\n").equals(varName));

System.out.print(name.equals(varName.trim()));
```

1. truetruetrue
2. falsetruefalse
3. falsefalsetrue
4. truefalsetrue
5. falsetruetrue
6. none of the above

Correct Answer: 5

Explanation:

varname has new line after 'Jane' that’s why first comparison will return false, so now three options 2,3,5 left. Second comparison will be true because we added to literal Jane '\n' (new line) character. Third comparison will return true because trim() method removes spaces new lines and tabs so variables will be equal.

- ➢ https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

45. What will be the output of the code below?

```
String  block= """
               Hello \
               World of-\
               \"""java 17\"""
               """;
 System.out.println(block);
```

1. Hello \
   World of-\
   """java 17"""

2. Hello \
   World of-\
   ""java 17""

3. Hello World of-""java 17""

4. Hello World of-"""java 17"""

5. Hello World of-
   """java 17""

6. code will not compile

Correct Answer: 4

Explanation:

\ - will escape new line and will help to escape triple double quotes so fourth option is valid.

- ➢ https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

Local Variable Type Inference

46. Look at the code below and select statements that are correct for local variable inference.

```
3. var greeting=null;
4. greeting = "Good morning";
5. final var reply="Good morning, too";
6. var answer = reply;
7. reply=null;
8. System.out.println(greeting+"-"+answer);
```

1. will not compile due to row 3.
2. will not compile due to row 7.
3. will print "Good morning-Good morning, too";
4. will print "Good morning-null";
5. using the final keyword with var is not allowed.
6. none of the above

Correct Answer: 1,2

Explanation:

Variable cannot be initialized with null, final variable value cannot be changed

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

47. Select statements that are correct about "var" type?

1. As the name states it is allowed in methods and constructors.
2. can be used with instance variable;
3. must be initialized
4. you don't need to initialize it explicitly when it is used with instance variable
5. variable of this type can be null, but only after correct initialization
6. Method or constructor can have variables of this type as a parameter.

Correct Answer: 1, 3, 5

Explanation:

1st is true - it can be used within methods and constructors only. 2nd is false - see answer of option 1. 3rd is true - it must be initialized with value so that compiler could infer its type. 4th is false - look at previous answer. 5th is true - var can be null but only after initialization which gave ability to compiler to infer type of var. 6th is false - parameters cannot be of type var because they'd not have value.

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

48. Select necessary steps that are required to make the class below valid.

```
3.  class Var {
4.  var var = "var";
5.  public void Var(var a) {
6.   var b=a;
7.   System.out.println(var);
8. }}
```

1. rename instance variable var because it is keyword
2. remove row 4
3. change type of instance variable var to String on row 4;
4. remove void keyword
5. change type to int of var a parameter on row 5;
6. remove row 6

Correct Answer: 3,5

Explanation:

We can't remove row 4 because it should output on row 7, so options 1,2 are invalid. Instance variable type can't be of type 'var', so if we need to change it then option 3 is valid. It is not necessary to remove keyword 'void' so option 4 is invalid, code will work with or without it. 5th options is valid, we need to change type of method parameter. Option 6 is not necessary.

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

Abstract Class

49. Select statements that are true about abstract class?

1. must have at least one abstract method
2. abstract method may or may not be implemented by subclasses
3. abstract methods functionality already realized in abstract class
4. Abstract class can be sealed class.
5. none of the above

Correct Answer: 4

Explanation:

Abstract class - can have no abstract method. If class has abstract method then it must be implemented in child class. It can be sealed class. Abstract class do not implement abstract method.

- https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

50. What will be the result of output?

```
public abstract class AbstractMan {
    public abstract String getGreeting();
    public void sayHello() {
```

```
System.out.print(String.valueOf(getGreeting()).trim
());
    }
}
...
public class ChineseMan extends AbstractMan{
    @Override
    public String getGreeting() {
        return "Ni Hao";
    }
}
...
public class EnglishMan extends AbstractMan{
    @Override
    public String getGreeting() {
        return "Hi";
    }
}
...main
AbstractMan em = new EnglishMan();
AbstractMan chm = new ChineseMan();
AbstractMan abs = new AbstractMan() {
    @Override
    public String getGreeting() {
        return null;
    }
};
em.sayHello();
chm.sayHello();
abs.sayHello();
```

1. code will not compile
2. HiNi Haonull
3. HiNi Hao
4. NullPointerException
5. HiNiHaonull

Correct Answer: 2

Explanation:

Code will output all greetings. First is 'Hi' then 'Ni Hao' and 'null'. There wouldn't be Nullpointer exception because String.valueOf(null) will convert it to string 'null'.

- https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Arrays and Collections

1. What will be the result of output?

```
public class Item {
   public String name;
   public int id;
   public Item(String name, int id) {
      this.name = name;
      this.id = id;
   }
}
 MapItem overrides equals() and hashCode()
public class MapItem {
   public String name;
@Override
public boolean equals(Object o) {
   MapItem mapItem = (MapItem) o;
   return Objects.equals(name, mapItem.name);
}
@Override
public int hashCode() {
   return Objects.hash(name);
}
}
--main method
var i1= new Item("Item", 1);
var i2= new Item("Item", 1);
Map<Item, Integer> map = new HashMap<>();
```

```
map.put(i1, 11);
System.out.println("map = " + map.get(i2));

Map<MapItem, Integer> map2 = new HashMap<>();
var m1= new MapItem("MapItem");
var m2= new MapItem("MapItem");
map2.put(m1, 12);
System.out.println("map2 = " + map2.get(m2));
```

1. map = 11, map2 = 12
2. map = null, map2 = 12
3. map = null, map2 = null
4. map = 11, map2 = null
5. none of the above

Correct Answer: 2

Explanation:

Map with Item as a key will return null when we call map.get(i2) because equals method compare their references. In second map we override equals method so that it compares their name field then MapItem with same name will be treated as a same key.

- ➢ https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- ➢ https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

2. Select statements that are true about collections?

1. Set relies on the equals() method of inserted Class to reveal duplicates.
2. When using Custom class as key in HashMap developer must override toString() method to make work with HashMap as expected.
3. HashMap can use primitives as a key.
4. String is good candidate to be key in HashMap
5. HashSet saves ordering of added items.

Correct Answer: 1,4

Explanation:

Set relies on equals method to exclude duplicates. When using Custom class as a Hashmap key developer must override equals and hashCode methods. HashMap cannot use primitives as a key. String is good candidate because it is immutable.Set does not saves order of added items

- ➢ https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- ➢ https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

3. Select statements that are true about unmodifiable collections?

1. In an unmodifiable collection we can add, remove and replace items.
2. List.of(), List.of(i1), List.of(i1¦) are methods to create an unmodifiable list.
3. Unmodifiable collection can not be modified except it stores items that can mutate.
4. Unmodifiable collections can not be modified in any cases.
5. Collections that are created using unmodifiable factory methods are space efficient.

Correct Answer: 2, 3, 5

Explanation:

As its name states unmodifiable collection cannot be change after creation. 2nd option is a factory methods of creation unmodifiable collections. 3rd is valid - unmodifiable collection that store mutable objects can be changed. 4th is false - see option 3. 5th is true - The collections returned by the convenience factory methods are more space efficient than their modifiable equivalents. Less objects are created.

- ➢ https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- ➢ https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

Concurrency

4. What is the result of the code below?

```
var threadOne = new Thread(() -> {
   System.out.print("threadOne");
});
var threadTwo = new Thread(() -> {
   try {
      Thread.sleep(1000);
   } catch (InterruptedException e) {
      throw new RuntimeException(e);
   }
   System.out.print("threadTwo");
});
threadTwo.setDaemon(true);
threadOne.start();
threadTwo.start();
System.out.print("main");
```

1. threadOnethreadTwomain
2. threadOnemain
3. threadOnemainthreadTwo
4. threadOnemain or mainthreadOne
5. threadTwothreadOnemain

Correct Answer: 4

Explanation:

Since threadTwo is daemon and it sleeps for 1 second it will not be printed because program will finish its execution. So all cases with threadTwo are invalid. Because we don't know order of thread execution 'main' and 'threadOne' can be printed in any order that's why option 4 is valid.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

5. Select statements that are true about Concurrency in java?

1. Runnable is the functional interface.
2. new Thread().run() and new Thread().start() methods make execution of code in another thread.
3. JVM will not wait until execution of daemon thread.
4. Constructors, methods of shared class can be synchronized to avoid unexpected behavior execution of programs.

Correct Answer: 1, 3

Explanation:

1st is true. 2nd is false - when we call new Thread().run() directly it will be executed in current thread. 3rd is true. 4th is false - Constructors cannot be synchronized.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

6. Select statements that are true about Concurrency in java?

1. Intrinsic Lock occurs when thread access synchronized method of class.
2. Locking the current class multiple times(calling another synchronized method) became enabled by reentrant synchronization.
3. Deadlock is a situation when one thread is waiting for another thread which owns class.
4. Starvation is when one thread waiting for response of second thread
5. Livelock is the case when a synchronized method takes too long for execution and all other threads are waiting for that method.

Correct Answer: 1,2

Explanation:

1st and 2nd are true. 3rd is false - Deadlock is when Thread A using resource B and Thread B waiting for resource A. In current situation they will wait forever. Starvation is usually occurs when thread execution occurs not in round-robin fashion. Some threads take long time for execution so the others will wait and this is called starvation. Livelock is similar to Deadlock when Thread A and Thread B tries to use same resource but finds that it busy and fails then tries again after same amount of time and again fail and it goes indefinitely.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

7. What do you need to change so that threads will increment only 1 count variable among 3 Count objects? Select the correct statement.

```
 Count.java
public class Count {
   public String name;
   public int count = 0;
   public void increase(String t) {
      count = count + 10;
   }
}

¦ main method

public static void main(String[] args) {
   Count countOne = new Count();
   Count countTwo = new Count();
   Count countThree = new Count();
   while(true) {
      new Thread(() -> countOne.increase()).start();
      new Thread(() -> countTwo.increase()).start();
      new Thread(() ->
countThree.increase()).start();
   }
}
```

1. make the increase() method synchronized.
2. make the increase() method static synchronized.
3. code will not compile.

4. make the count variable static.
5. change Count class as public static class Count.

Correct Answer: 4

Explanation:

In code sample we created 3 instances of Count so synchronization of method will not help to increment single variable. We need to make field count static because if its nature there will be single static field among multiple objects. We cannot create such class 'public static class Count' because static class is usually inner class but not top-level class.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Control Flow (Loop, switch, if/else)

8. Select statements that are true about the latest switch statement.

1. default value is not mandatory
2. default value is not mandatory when switch works with enums
3. break value became optional
4. old syntax of switch is not valid
5. yield is used to return result
6. none of them

Correct Answer: 2, 3, 5

Explanation:

Default value is still mandatory. Break is optional. Old syntax is still valid and can be used. Yield is used to return value.

- https://docs.oracle.com/en/java/javase/17/language/switch-expressions.html

Date/Time API

9. Select statements and put them in a blank space that will help to minus one year and one day from LocalDate.now() when it is leap year and output a new date.

```
var date = LocalDate.of(2024, Month.FEBRUARY, 29);
if(date.isLeapYear()) {
   if(date.equals(LocalDate.of(date.getYear(), 2, 29))) {

        // blank space

       System.out.println("newDate:" + newDate);
   }
}
```

1.
```
 var period = Period.ofYears(1).ofDays(1);
       var newDate = date.minus(periodYear);
       outputs newDate: 2023-02-28
```

2.
```
 var periodYear = Period.ofYears(1);
       var periodDay = Period.ofDays(1);
       var newDate =
date.minus(periodYear).minus(periodDay);
                 outputs newDate: 2023-02-27
```

3.
```
var periodYear = Period.ofYears(1);
 var periodDay = Period.ofDays(1);
 var newDate =
date.minus(periodYear).minus(periodDay);
 outputs newDate: 2023-02-28
```

4. var newDate = date.minus(1, ChronoUnit.YEARS).minus(1, ChronoUnit.DAYS)
   outputs newDate: 2023-02-27
5. none of the above

Correct Answer: 2,4

Explanation:

1st is false - there is no such variable periodYear. 2nd is true - when we minus 1 year it will be 2023 that is not leap so February will contain 28 days, then we minus 1 day so result will be 2023-02-27. 3d is false because day value is 28. 4th is correct.

- https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

10. Consider the scenario below. Find the result of output.

```
var date = LocalDate.of(2024, Month.FEBRUARY, 29);
if(date.isLeapYear()) {
   if(date.equals(LocalDate.of(date.getYear(),
Month.FEBRUARY, 29))) {
      var d1 = date.minus(1,
ChronoUnit.YEARS).minus(1, ChronoUnit.DAYS);
      var d2 = date.minus(1,
ChronoUnit.DAYS).minus(1, ChronoUnit.YEARS);
      var d3 = date.minus(1,
ChronoUnit.YEARS).plus(1, ChronoUnit.DAYS);
      System.out.println(d1 + " " + d2 + " " + d3);
   }
}
```

1. 2023-02-27 2023-02-28 null
2. 2023-02-28 2023-02-28 2023-02-28
3. 2023-02-27 2023-02-27 2023-03-01
4. 2023-02-27 2023-02-28 2023-03-01
5. 2023-02-28 2023-02-28 2023-03-01

Correct Answer: 4

Explanation:

If we minus 1 day from date firstly and then year result will be 2023-02-28. When we minus 1 year from date and then 1 day then result will be 2023-02-27. When we minus 1 year and add 1 day then result will be 2023-03-01. So result will be 4. Others are invalid

- https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

Exception Handling

11. Select statements that are true about exceptions in java.

1. finally block executed even though there are unexpected exceptions thrown in try block.
2. try with resource release resources when they are not needed implicitly
3. if JVM exits while executing a try catch block then finally block will not be executed.
4. Error is thrown in the program when hardware problems occur.
5. None of the above

Correct Answer: 1,2,3,4

Explanation:

finally block executed even on unexpected exceptions. try with resource works with classes that implements Autoclosable so resource will be closed properly implicitly when not used. In case when jvm exits finally block will not be executed. Error is thrown on hardware problems.

- ➢ https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

12. Select correct variants of execution of code below .

```
var random = Math.random() * 10;
try {
   if((int)random % 2 == 0) {
      throw new SQLException();
   } else {
      throw new IndexOutOfBoundsException();
   }
} catch (SQLException e) {
}
```

1. code will not compile.
2. it will work without exceptions all the time.
3. In some cases there can be unhandled exceptions.
4. If SQLException is thrown then the program will be exited.
5. IndexOutOfBoundsException will never occur because we do not use arrays here.

Correct Answer: 3

Explanation:

Because we are using random() numbers there can be case when value of random will be odd number. So code will throw IndexOutOfBoundsException which is not handled in catch clause.

- ➢ https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

13. Select valid usage of exception handling(try catch, try with resource).

1. try() with resource() {}
2. try (FileWriter writer = new FileWriter("")) {}
3. try (Scanner sc = new Scanner(System.in)) {System.out.println("sc.nextLine() = " + sc.nextLine());}
4. try (FileWriter writer = new FileWriter("file.txt"); Scanner sc = new Scanner(System.in)) {System.out.println("sc.nextLine() = " + sc.nextLine());}
5. FileReader read = new FileReader("content.txt"); try {read.read();} catch (Exception e) {}finally {read.close();}

Correct Answer: 2,3,4,5

Explanation:

1st is false - no such syntax like 'with resource'. Other variants are valid usage of try catch and try with resources.

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Garbage collection

14. Which of the following is true about garbage collection?

1. Calling System.gc() will clean old objects instantly
2. Objects that were created without reference are eligible for garbage collection
3. Objects that were created in the method that is already out of scope are eligible for garbage collection
4. Explicitly calling System.gc() can™t run garbage collection directly
5. Objects that are eligible for garbage collection can stay in a heap until the end of execution of the program
6. gc clean old objects from heap and stack memory.

Correct Answer: 2,3,4,5

Explanation:

System.gc() - when we call this method jvm do not guarantee instant execution of it. Objects that are lost it's reference variables are eligible for gc. When object goes out of scope it is eligible for gc. Garbage collector clean objects that lost its references from heap.

- https://blogs.oracle.com/javamagazine/post/understanding-garbage-collectors

Java OOA instanceof

15. Select statements that are correct alternatives of code below using pattern matching.

```
      if(list instanceof ArrayList) {
          ArrayList arrayList = (ArrayList) list;
          if(arrayList.size() > 0) { System.out.println("is
not empty"); }
      }
```

1.
   ```
   if(list instanceof ArrayList arrayList ||
   arrayList.size() > 0) {
      System.out.println("is not empty");
   }
   if(list instanceof AbstractList arrayList &&
   arrayList.size() > 0) {
      System.out.println("is not empty");
   }
   ```

2.
   ```
   if(list instanceof final ArrayList arrayList ||
   arrayList.size() > 0) {
      System.out.println("is not empty");
   }
   ```

3.
   ```
   if(list instanceof final ArrayList arrayList &&
   arrayList.size() > 0) {
      System.out.println("is not empty");
   }
   ```

4.
   ```
   if(list instanceof final ArrayList arrayList) {
      arrayList = (ArrayList)list;
      if(arrayList.size() > 0) { System.out.println("is not
   empty cast"); }
   }
   ```

Correct Answer: 2,4

Explanation:

1st is invalid - we need to use here && if list is instance of ArrayList only then we can check its size if we use || then code will not compile. 2nd option is valid case. 3rd is false see answer 1. 4th is true. 5th is false because we are trying to reassign final variable.

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-instanceof-operator.html

Java OOA Overloading, Overriding, Interfaces

16. What will be the output of the code below?

```
public class Main {
   public Main(int i) {System.out.print("int");}
   public Main(double d)
{System.out.print("double");}
   public Main(float f) {System.out.print("float");}
   public static void main(String[] args) {
      var var = 10/3;
      new Main(var);
      var = (int)var;
      new Main(var);
      new Main(var*(2f)/2);
   }
}
```

1. doubleintfloat
2. doubledoublefloat
3. intintdouble
4. intintint
5. intintfloat
6. none of the above

Correct Answer: 5

Explanation:

Answer is 5 option. When we divide int and int then result will be also integer. So first and second output will be 'int'. Then after multiplication with float third output will be float. So result will be 'intintfloat'.

➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

17. What will be the output of the code below when run new Child()?

```
public class Parent {
   public String label = "parent";
   public String name = "iamparent";
   public int year = 2021;
}
class Child extends Parent {
   public String label = "child";
   public int year = 2022;
   public Child() {
      var result = this.label + super.label + super.year
+ this.name;
      System.out.print(result);
   }
}
```

1. childparent2021null
2. childparent2021iamparent
3. parentparent2021iamparent
4. parentparent2021null

5. code will not compile

Correct Answer: 2

Explanation:

Result will be 'childparent2021iamparent'. Keyword this is used to reference local if not exist than parent fields. So this.label will return child and this.name will return iamparent. Keyword super is used to reference parent fields and methods. Super.label will return parent and super.year is 2021

- ➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

18. Select statements that are true about Constructors in java.

1. at least one Constructor must be created in class.
2. class can have multiple Constructors with different parameters.
3. class can have private Constructors.
4. classes with private Constructors can be instantiated because of the default constructor that was created by java.
5. Constructor can have var as parameter

Correct Answer: 2,3

Explanation:

Java class can live without explicitly created constructor. Java class can have multiple constructors with different parameters due to polymorphism. Java class can have private Constructor. Constructor parameter cannot be of type var because var must be initialized.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

19. What will be the output of the code below?

```
public class Machine {
   public String generalType = "Land";
   public String name ="machine";
   public int power = 200;
}
public class Car extends Machine{
   public String type = "Sedan";
   public String name = "ElectricalCar";
   public int power = 250;
}
var car = new Car();
System.out.println(car.name + car.type +
car.generalType + car.power);
var machine = new Machine();
machine = new Car();
System.out.println(machine.name +
machine.generalType + machine.power);
```

1. code will not compile

2. ElectricalCarSedanLand250
   machineLand200

3. ElectricalCarSedanLand200
   machineLand200

4. ElectricalCarSedanLand250
   ElectricalCarSedanLand250

5. machineLand200
   machineLand200

Correct Answer: 2

Explanation:

Code will compile, output will be variant 2. Car is instance of class Car so output will be fields of class Car except field 'generalType' it's taken from parent. var machine will be type of Machine class. So output will be fields of Machine class.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

20. What will be the output of the code below?

```
public class GrandFather {
   public String name = "GrandFather";
   public GrandFather() {System.out.print(name);}
}
..
public class Father extends GrandFather{
   public String name = "Father";
   public Father() {System.out.print(name);}
}
..
public class Child extends Father{
   public String name = "Child";
   public Child() {System.out.print(name);}
}
...
GrandFather obj = new Child();
System.out.print(obj.name);
```

1. ChildFatherGrandFatherChild
2. ChildFatherGrandFatherChildGrandFather
3. FatherFatherChildGrandFather
4. GrandFatherFatherChild
5. GrandFatherFatherChildGrandFather

Correct Answer: 5

Explanation:

Since Child extends Father and Father extends GrandFather java will load them also. So output firstly will be 'GrandFather' then 'Father' than 'Child'. Last output will be the name of GrandFather that is 'GrandFather' - so answer will be 5th option.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

JDBC

21. Select statements that are true about JDBC.

1. JDBC supports hql(hibernate query language).
2. Statement used with query string only.
3. Statement can have parameters.
4. PreparedStatement can have parameters.
5. CallableStatement runs the query asynchronously.
6. None of the above.

Correct Answer: 2,4

Explanation:

JDBC does not support HQL which is hibernate query language. 2nd is true. 3rd is false - Statement do not have parameters. 4th is true PreparedStatement can have parameters.CallableStatement used to call stored procedures.

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

22. Select statements that are true about JDBC.

1. CallableStatement used to run stored procedures.
2. CallableStatement runs the query asynchronously.

3. When the same query runs multiple times with Statement then it gets remembered by the database for performance.
4. Sql injection can occur while using the Statement
5. try(var prepStatement = con.prepareStatement()) {} - will compile
6. none of the above

Correct Answer: 1,4

Explanation:

CallableStatement used to call stored procedures. When same query runs with PreparedStatement then it gets remembered by the database for performance. Sql inject is possible with Statement because it works with plain string query. 5th option is invalid - it needs query as parameter. 6th is invalid

- https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Lambda and Functional Interface

23. Select statements that are true about lambda and functional interface.

1. Interface with a single private method is a valid functional interface.
2. Interface with method toString and return type String is valid functional interface.
3. Interface with method toString and return type int is valid functional interface.
4. Interface with method to toString() with parameter and return type String - is valid functional interface.
5. Interface with method toString() with return type String and also with any public void method is a valid functional interface .

Correct Answer: 4,5

Explanation:

Functional interface must have exactly one abstract method and it should not be java.lang.Object methods. 1st is invalid - Private method is not abstract. 2nd is invalid - Interface with toString method and String as a return type is not Functional Interface by definition. 3rd is invalid - interface can't have toString() method with return type int because it clashes with Objects method toString(). 4th is true toString(int i) with parameter do not clash with Objects method and will be treated as abstract method. 5th is true - Interface with public void method is treated as Functional interface. 5th is true –

- https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

Localization

24. Select correct output of the code below.

```
Locale.setDefault(Locale.US);
var num = 12_500_000;
var cfi =
CompactNumberFormat.getCompactNumberInstance();
var cflong =
CompactNumberFormat.getCompactNumberInstance(Locale.getDefault(),
NumberFormat.Style.LONG);
var nform = NumberFormat.getNumberInstance();
System.out.print(cfi.format(num) + "/");
System.out.print(cflong.format(num) + "/");
System.out.print(nform.format(num));
```

1. 12M/12 million/12,500,000
2. 12,5 M/12,5 million/12,500,000
3. 12.5M/12.5 million/12,500,000
4. 12,5M/12,5 million/12.500.000
5. 12-M/12-million/12_500_000
6. none of the above

Correct Answer: 1

Explanation:

CompactNumberFormat rounds number in HALF_EVEN format. In out case 12.5 will be rounded to 12 because 13 is odd. So option with 12M/12 million/12,500,000 is correct one.

- https://docs.oracle.com/javase/tutorial/i18n/

25. Select statements that are true about Localization in java.

1. enUs is valid Locale identifier
2. NumberFormat, ResourceBundle, Locale are classes that used to help with localization of program
3. date, currency output style value can vary due to type of Locale
4. ResourceBundle works with property files that contain localizations.
5. ResourceBundle works with database tables that store localizations.

Correct Answer: 2,3,4

Explanation:

enUs is incorrect Locale format. 2nd is valid. 3rd is valid - date, currency format style can be different in us or hindi locales. Resource bundle works with files and not database tables that store translations.

- https://docs.oracle.com/javase/tutorial/i18n/

Math API

26. What will be the output of the code below?

```
      double d = 3.79;
       var ceil = Math.ceil(d);
       var floor = Math.floor(d);
       var round = Math.round(d);
       System.out.println((round == ceil)+(round ==
floor)+(ceil == floor) + "");
```

1. truetruefalse
2. falsefalsefalse
3. falsetruetrue
4. code will not compile
5. truetruetrue
6. none of the above

Correct Answer: 4

Explanation:

code will not compile because we can"t sum boolean values.

- https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Modules

27. Select statements that are true about java modules.

1. In the latest versions of java it is mandatory to use modules.
2. java.base is a core module that other modules rely on.
3. exports and open keywords are used with package names but keywords require used with module names.
4. requires modifiers are: private, transitive, static.
5. exporting package is also applied to its subpackages.
6. none of the above

Correct Answer: 2,3

Explanation:

modules are not mandatory. java.base is core module and other modules rely on it. 3rd option is true. 4th is false - private is incorrect modifier for requires. 5th is false - subpackages are not exported.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

28. Select statements that will help to run modules.

```
1 //JavaVersion.java
2 package m.jversion;
3 public class JavaVersion {
4 public void javaVersion()
{System.out.println("Java version is 17");}}
5 //module-info.java
6 module jversion
7
8 //Main.java
9 package m.main;
10 import JavaVersion;
11 public class Main {
12 public static void main(String[] args) {new
JavaVersion().javaVersion();}}
13 //module-info.java
14 module main
15 //modules located in mymodules folder;
```

1. change line 9 to jversion;
2. change line 10 to import m.jversion.JavaVersion
3. change line 6 to module jversion {}
4. change line 14 to module main {requires jversion; }
5. change line 6 to module jversion { exports m.jversion;}
6. change line 6 to module jversion { exports jversion;}

Correct Answer: 2,4,5

Explanation:

in order to use JavaVersion class it's module must be exported as in option 5. Then main module must require it as in option 4. Then correct import as in option 2.

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

29. Select statements that are true about modules. (multi-select)

1. java.base module is only package that does not need to require explicitly
2. all core java modules are not needed to require explicitly in java program
3. module name can be used in exports and open keywords
4. module require other modules
5. modules add new level of encapsulation
6. none of the above

Correct Answer: 1, 4, 5

Explanation:

java.base is only core java module that is loaded implicitly, other needs to be required. Keyword exports used with package name. 4th and 5th are valid.

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

30. Consider having such modules as utility and main. Select statements that will show correct usage of modules.

1. if modules are in same project and main called method from utility, no need to code requires utility;
2. name of module be the same as name of root folder where module-info.java
3. utility does not need to export its packages for modules inside same project
4. If main uses methods from utility module then module-info.java must have code - requires utility;
5. if most of packages are concealed no need to create module-info.java

Correct Answer: 2,4

Explanation:

To use Class methods from module utility, we must add 'require utility' in module-info.java of main module, even if they are in same project. Name of module must be same as root folder where module-info.java located. If most packages are concealed we need to export open packages.

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

Arithmetic/Boolean Expressions (Operators, Promotion, Casting)

31. What is the result of the following code snippet

```
boolean bool = false;
 float a = 10;
 float b = 10.0f;
 double d = 20;
 if(bool=true) {
     System.out.print("d++");
     d++;
 }
  float c = (float)d + b ++;
  System.out.print(c*(++a));
```

1. d++342

2. 341

3. 330

4. d++341.0

5. 330

6. code will not compile

Correct Answer: 4

Explanation:

Code will compile. In if condition we set value of bool to true so when true it will print 'd++' and increase value of d to 21.0. Then 21+10 (postincrement of 'b' will return number and then increment) will give us c = 31.0. Then 31.0 * 11.0(preincrement will increment and then return number) will give us 341.0. Final result will be d++341.0

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

32. What is the data type of variables c, f, g?

```
int a = 10;
long b = 5;
var c = a + b;
double d = 20.0;
float e = 11.0;
var f = d + e;
var g = (short)10 * (short)5;
```

1. int, double, int
2. long, double,short
3. long, double, int
4. int, float, short
5. long, float, short
6. none of the above

Correct Answer: 6

Explanation:

When we initialize float with decimal we need to add 'f' so it look like float e = 11.0f otherwise compiler will treat that we assigning double value to float variable and throw exception.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

33. What is the data type of variable result and what is the value of result?

```
short x = 124;
byte xy = 5;
var sum = (x+xy)/2f;
var result = (int)2.5 + sum;
System.out.println(result);
```

1. float, 66.5
2. double, 66.0
3. int, 66
4. double, 66.5
5. none of the above

Correct Answer: 1

Explanation:

sum of x+xy = 129(int) will be 64.5 of type float when divided by 2f. Final result will be 2 + 64.5 = 66.5 of type float. By the rule of java math operations result value will converted to bigger type. In our case float is bigger than int.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

Record

34. Select statements that are true about record.

1. record class cannot extend sealed class
2. record fields may be private or public
3. explicitly create constructor with additional logic
4. since record is immutable you cannot override equals() and hashCode() methods
5. record can have custom methods
6. record can have instance variables, and static methods

Correct Answer: 3,5

Explanation:

Record is immutable, can extend sealed class and has private instance fields with public getter methods. It can override equals() and hashCode() methods and can have it's own Contructors and methods. You cannot declare instance variables (non-static fields) or instance initializers in a record class

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

35. Select statements that are true about record.

1. record class can have static instance fields
2. record can have native method
3. record cannot have inner class as usual java class
4. has private methods and private final fields
5. new instance of record class can be created using new keyword
6. record class has private final fields and is mutable

Correct Answer:

Explanation:

Record cannot have native method, but can have static instance fields and inner class. It has private final fields and public access methods. They can be created using new keyword. Record class is immutable.

- https://docs.oracle.com/en/java/javase/17/language/records.html

Sealed Classes

36. Select statements that are true about sealed classes.

1. sealed classes may not have child classes
2. sealed class may not use keyword permit if its child classes are located in the same file as parent class.
3. if child is located in same file as a parent sealed class then it can be without modifier such as final, sealed, non-sealed
4. child classes can not have its own child classes
5. none of the above

Correct Answer: 2

Explanation:

Sealed class must have child classes. If child classes located in same file we can omit permits keyword. Child classes must have one of this modifiers: final, sealed, non-sealed. Child classes can have their own child classes.

- ➢ https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- ➢ https://openjdk.org/jeps/411

37. Select statements that are true about sealed classes.

1. sealed class feature came to replace final keyword
2. sealed interface can permit record class
3. sealed class and its child classes must locate in same module if sealed class in named module
4. child classes must be in same package if sealed class in unnamed module
5. sealed interface can not permit record class
6. sealed interface can permit record class if it has modifiers sealed, non-sealed

Correct Answer: 2,3,4

Explanation:

1st is false - sealed class came to restrict inheritance mechanism. Sealed interface can permit record. Option 3,4 are also valid due to definition. 5th is false, 6th is false - record class can not be extended thats why keyword sealed is not allowed in records.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- https://openjdk.org/jeps/412

Streams

38. Consider the case when you have a list of Country objects. Country object has 2 instance fields: name of type String and cities is the list of city names.

Task is to get such cities that start with letter 'm' then make that name in upper case form and exclude duplicates if they exist. Select all possible variants.

1. countries.stream().flatMap(items -> items.cities.stream()).filter(i -> i.startsWith("m")).distinct();

2. countries.stream().flatMap(items -> items.cities.stream()).filter(i -> i.startsWith("m")).map(i -> i.toUpperCase()).distinct().collect(Collectors.toList());

3. countries.stream().filter(i -> i.startsWith("m")).map(i -> i.toUpperCase()).distinct().collect(Collectors.toList())

4. countries.stream().map(i -> items.cities.stream()).filter(i -> i.startsWith("m")).map(i -> i.toUpperCase()).distinct().collect(Collectors.toList());

5. countries.stream().flatMap(items -> items.cities.stream()).filter(i -> i.startsWith("m")).map(i -> i.toUpperCase()).collect(Collectors.toSet());

Correct Answer: 2,5

Explanation:

1st is invalid - needs to convert to uppercase and terminal operation. 2nd is valid - they use flatmap to loop over cities array. It filters and converts to uppercase and removes duplicates. 3rd is invalid they are not using flatmap. 4th is false - no flatmap. 5th is valid - they use flatmap, filters, converts to uppercase then collect as a set.

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

39. Select the correct value of the numberFirst variable.

```
var numbers = Stream.of(-23,32, 12,10,21,3, 40);
var numberFirst = numbers
      .filter(i -> i % 2 == 0).sorted().findFirst();
```

1. 10

2. 40

3. Optional[10]

4. 12

5. Optional[0]
6. none of the above

Correct Answer: 3

Explanation:

sorted array of even numbers will be 10, findFirst() method will return Optional of first element. So result will be Optional[10]

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

40. Select the correct value of the first variable.

```
var numbers = Stream.of(-2,312, 12,10,21,3, 22,21);
var first = numbers.sorted(Integer::compare).filter(i
-> i > 10).findFirst().get();
```

1. 20
2. 12
3. 312
4. 22
5. 21
6. none of the above

Correct Answer: 2

Explanation:

sorted array of numbers that are greater than 10 will be 12, findFirst().get() method will return 12.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

41. Convert list to map using streams. Select statements that are valid. var list = List.of(1,2,3,4,5,6,6);

1. list.stream().collect(Collectors.toMap(k -> k, v -> v/2));
2. list.stream().collect(Collectors.toMap(k -> k, v -> v));
3. list.stream().distinct().collect(Collectors.toMap(k -> k, v -> v/2));
4. list.stream().distinct().collect(Collectors.toHashMap(k -> k, v -> v/2));
5. list.stream().distinct().sorted().collect(Collectors.toMap(k -> k, v -> v/2));
6. list.stream().distinct().filter(i -> i < 5).collect(Collectors.toMap(k -> k, v -> v));

Correct Answer: 3,5,6

Explanation:

Since array contains duplicates variant 1, 2 will give us exception. Option 3 is valid. 4th option using toHashMap which is not exists. 5, 6 are valid usage

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

42. What is the value of the result?

```
var list = List.of(1,6,7,2,3,3,4,5,8,4);
var result = list.stream().distinct().filter(i ->
i%3==0).reduce(1, (c, d) -> c + d);
```

1. 23
2. 10
3. 13
4. 8
5. 12
6. 15

Correct Answer: 2

Explanation:

after filtration array will contain 3,6. reduce method with identity 1 will add 1+ 3+6 = 10. So result will be 10

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

43. What is the value of the result?

```
var array2d = List.of(List.of(0,2,4), List.of(3, 6, 9));
var result =
array2d.stream().flatMap(Collection::stream).filter(f
-> f%2!=0).reduce(0, (x, z) -> x+z);
```

1. 15

2. 13
3. 11
4. 12
5. 9
6. 8

Correct Answer: 4

Explanation:

After filtration array will contain 3,9 so their sum will be 12. Identity of reduce method is 0 so final result will be 12

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

Text Block

44. Which code will produce text Lorem Ipsum dolar?

1.
   ```
   String dummy = """
               Lorem %s %s""".formatted("Ipsum", "dolar");
   ```

2.
   ```
   String dummy = """
               Lorem %s %s""";
            dummy = dummy.formatted("Ipsum",
   "dolar");
   ```

3.
   ```
   String dummy = """
             Lorem %s %s""";
              dummy = dummy.format("Ipsum", "dolar");
   ```

4.
   ```
   String dummy = """
             Lorem %s
   %s""".setParam("Ipsum").setParam("dolar");
   ```

5.
   ```
   System.out.printf("Lorem %s %s", "Ipsum", "dolar");
   ```

Correct Answer: 1,2,5

Explanation:

option 1,2 uses String's formatted method to pass parameters. 3rd is false - no such format method of String. 4th option is invalid no such setParam method. 5th is true - java standard string format output

- https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

45. Select statements that are true about the text block

1. intend(4) will add 4 spaces to text block rows
2. format("Param1") will set parameter to text block containing "%s" symbol
3. text block can not be initialized in single code line;
4. syntax of text block """"Some string"""";
5. dynamic parameter can by set using "%s" symbol and formatted() method
6. none of the above

Correct Answer: 1,3,5

Explanation:

1is is true intend method add spaces to text block rows. 2nd is false - format() no such method. 3 is true - text block will take at least 2 lines. 4th is false - see answer 3. 5th is valid - we can add parameters using formatted() method.

- ➢ https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

46. What is the output?

```
String a = "",b = "";
var aa = new StringBuffer(); var bb = new
StringBuffer();
System.out.println(a.equals(b) + "-" +
aa.equals(bb) + "-" + a.equals(aa));
```

1. true-true-false
2. true-false-false
3. true-true-false
4. true-true-true
5. false-true-false
6. true-false-true

Correct Answer: 2

Explanation:

compare of literals will give us true. Compare StringBuffer objects will give us false because it compares its references. We need to call toString() method explicitly to get String value. Compare of literal with object will give us false

- https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

Local Variable Type Inference

47. Select statements that are correct about "var" type?

1. used to simplify code and make it more readable.
2. can be used with primitives and String only
3. can change its type dynamically in runtime
4. variable, method and package name can be named as   var
5. class or interface can be named as var

Correct Answer: 1,4

Explanation:

Var- local variable inference came to make java code more readable and make initialization process more convenient. It can be used with primitives and objects. It's type cannot be changed after initialization. We can't name class or interface as a 'var' but can use it as variable, method or package name.

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

48. Select only necessary steps that are required to make the class below valid.

```
3. class var {
4.  String outside;
5.  public void sayVar() {
6.    var a=outside;
7.    String internal;
8.    var b = internal
9.    System.out.println(a);
10.   System.out.println(b);}}
```

1. change name of the class to valid one
2. initialize variable on row 4
3. initialize variable on row 7
4. change type from String to var on row 4
5. do not need to modify the class, it is already valid.

Correct Answer: 1,3

Explanation:

We can't name class as var, so need to change it and apply option 1. Line 6 is valid because instance variable outside has default value that is empty string. Line 8 is invalid because internal variable needs to be initialized so apply option 3. Then code is valid.

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

49. Select only necessary steps that are required to make the code compile?

```
package var;
public class Var {
 public void sayHi(boolean sayHiOrNot) {
    String label;
    if(sayHiOrNot) { label = Hi; }
    var hi =  label;
    System.out.println(hi);
}}
```

1. change name of package to valid one;
2. change name of the class to valid one;
3. code is already valid and will output hi on new Var().sayHi(true)
4. add else statement with initialization of variable label
5. none of the above

Correct Answer: 4

Explanation:

Label in some cases wouldn't be initialized that’s why compiler will throw error so we need to add else statement where we also initialize variable label

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

Abstract Class

50. What will be result of code below?

```
public abstract class Sample {
   public abstract void sampleMethod();
}
...
public class ChildOne extends Sample{
   public ChildOne() { sampleMethod();}
   @Override
   public void sampleMethod() {
System.out.print("sampleMethod()");}
}
...
public abstract class ChildTwo extends Sample{
   public ChildTwo() {System.out.print("ChildTwo");}
}
...
public class ChildThree extends ChildTwo{
   public ChildThree() {sampleMethod();}
   @Override
   public void sampleMethod()
{System.out.print("ChildThree");}
}
...main
Sample s1 = new ChildOne();
Sample childTwo = new ChildThree();
```

1. sampleMethod()ChildThree
2. code will not compile

3. sampleMethod()ChildTwoChildThree
4. sampleMethod()ChildTwo
5. sampleMethod()nullChildThree
6. sampleMethod()ChildThreeChildTwo

Correct Answer: 3

Explanation:

first output will be sampleMethod(), second output will be 'ChildTwo' because it's parent of ChildThree and java load all parents of child class. Third output will be ChildThree.

- ➢ https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Arrays and Collections

1. Select statements that are true about an unmodifiable set, list, map?

1. collection considered to be thread safe even if it contains mutable elements.
2. Allows to create a collection with null value.
3. collections are serializable if all its items are serializable.
4. List.of(), List.copyOf() used to create an unmodifiable set.
5. Collections.unmodifiableMap(randomMap) is more memory efficient Map.of(k1, v1, k2, v2¦)

Correct Answer: 3,4

Explanation:

Unmodifiable collection considered to thread safe if it contains immutable objects. Null values are not allowed. Collections are serializable if all its items are serializable. Option 4 is factory methods to create Unmodifiable collections. 5th is false - collection created by factory methods are more space efficient because less objects created.

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html

- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

2. What will be the result output of the code below?

```
Set<String> s = new TreeSet();
s.add("9");s.add("19");
s.add("112");s.add("3");
int counter = 0, amount = 0;
for(String str:s) {
   if(counter >=2 ) {break;}
   amount = amount + Integer.parseInt(str);
   counter++;
}
System.out.println(amount);
```

1. 28
2. 12
3. 22
4. 131
5. 115

Correct Answer: 4

Explanation:

TreeSet will order items in natural order [112,19, 3, 9], so sum of first 2 elements is 131

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html

- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

3. What will be the result output of the code below?

```
Set<String> hashSet = new HashSet<>();
hashSet.add("9");hashSet.add("19");
hashSet.add("112");hashSet.add("3");
int counter = 0, amount = 0;
for(String str:hashSet) {
   if(counter >=2 ) {break;}
   amount = amount + Integer.parseInt(str);
   counter++;
}
System.out.println(amount);
```

1. 21
2. Output result will rely on how HashSet stores their elements.
3. 116
4. 132
5. 117

Correct Answer: 2

Explanation:

HashSet does not order items and not save order how item was added, that's why we cannot predict result.

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html

- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

4. What will be the result output of the code below?

```
Set iset = new TreeSet();
iset.add(8);iset.add(15);
iset.add(32);iset.add(11);
int counter = 0, amount = 0;
for(Integer i:iset) {
   if(counter >=2 ) {break;}
   amount = amount + i;
   counter++;
}
System.out.println(amount);
```

1. 19
2. compile error
3. 23
4. 40
5. 47

Correct Answer: 2

Explanation:

Set's generic type is Object then row with for(Integer i:iset) {} will throw compile error.

- https://docs.oracle.com/en/java/javase/17/core/creating-immutable-lists-sets-and-maps.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

Concurrency

5. Consider the case below and select a variant that could lead to deadlock.

```
public class SyncClass {
    static String resourceOne = "resourceOne";
    static String resourceTwo = "resourceTwo";
    public SyncClass() {
    }
    public void useResourceOneAndTwo() {
        synchronized (resourceOne) {
            System.out.println("method 1and2:
resourceOne =: " + resourceOne);
            synchronized (resourceTwo) {
                System.out.println("method 1and2:
resourceTwo = " + resourceTwo);
            }
        }
    }
    public void useResourceTwoAndOne() {
        synchronized (resourceTwo) {
            System.out.println("method 2and1:
resourceTwo = " + resourceTwo);
            synchronized (resourceOne) {
                System.out.println("method 2and1:
resourceOne = " + resourceOne);
            }
        }
    }
}
¦.
main method
SyncClass sc = new SyncClass();
```

1. new Thread(() -> {
   sc.useResourceTwoAndOne();}).start();
2. new Thread(() -> {
   sc.useResourceTwoAndOne();}).start();
   new Thread(() ->
   {sc.useResourceOneAndTwo();}).start();
3. new Thread(() -> {
   sc.useResourceTwoAndOne();
   sc.useResourceOneAndTwo();}).start();
4. new Thread(() -> { Thread.sleep(1000);
   sc.useResourceTwoAndOne();}).start();
   new Thread(() ->
   {sc.useResourceOneAndTwo();}).start();
5. none of the above

Correct Answer: 2

Explanation:

Deadlock can occur when we run 2 different threads (A and B) and A will call method useResourceOneAndTwo() and B will call useResourceTwoAndOne(). This 2 methods use same resources in synchronized block. When thread A acquires resourceOne and thread B acquires resourceTwo at the same time after that each of them will request for next resource which is still in use and can not be released. So deadlock occurs. We need at least 2 threads. We have only 2 options that run 2 threads they are 2, 4. Option 4 will not cause deadlock because one of threads will sleep for 1 second so the second thread will finish its execution.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

6. Consider a case when there is a need to accomplish a big task and to do it effectively the program should divide a task recursively into smaller tasks until it can be done and completed fast. Select the best variant that will help to achieve it .

1. Use fork/join framework.
2. Use loops and threads.
3. Use future/callable
4. Use do while to divide task in single thread
5. Use streams

Correct Answer: 1

Explanation:

Fork/join framework is used to execute complex task by dividing it recursively into smaller and completable small tasks. This is best option because it's a main goal of framework.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

7. Select statements that are true about concurrency in java.

1. Thread pools are good because creation of new thread for each request is time consuming.
2. Immutable objects can be used in multithreaded programs without worrying about thread safety.
3. AtomicInteger is thread-safe.
4. Callable interface do not return value.
5. Thread interference can not be solved with synchronized blocks.

Correct Answer: 1,2,3

Explanation:

1st is true - Thread Pools are more efficient because creation of new threads on every call takes time. 2nd, 3rd are true - immutable objects and AtomicInteger are thread safe. First one is simply will not be changed by multiple threads and second was designed to work in concurrent environment. 4th is false - Callable interface return result in async fashion. 5th is false - synchronization helps with thread interference. When one thread enters synchronized block other threads will wait until it finishes.

- ➢ https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

8. Fill in the blanks.

___ - is a situation which occurs when threads are executed due to their priority.

___ - is the case when jobs that are expected to be done in an ordered manner, executed at the same time in critical section.

1. Livelock, Thread starvation.
2. Deadlock, Liveness
3. Thread starvation, Race condition.
4. Intrinsic Lock, LiveLock
5. Reentrant Lock, Race condition

Correct Answer: 3

Explanation:

1st blank is - Thread starvation is when one thread can be executed for a long time and others are waiting. 2nd is Race Condition when threads that are executing in critical section try to modify some common variable at same time which will lead to different work of a program rather when they are acting sequentially.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Control Flow (Loop, switch, if/else)

9. Select statements that will fix the code below.

```
public enum EvenNumber {
   TWO, FOUR, SIX, EIGHT
}
var evenNum = EvenNumber.TWO;
int result = switch (evenNum) {
   case TWO -> 2; break;
   case FOUR -> {
      return 4;
   }
   case SIX: {
      yield 6;
   }
};
```

1. remove yield from case SIX; add return 6;
2. add case EIGHT
3. in case FOUR replace keyword return to yield
4. case SIX change symbol : to arrow ->
5. without default case it will not work, even if we added case EIGHT
6. case TWO remove keyword break

Correct Answer: 2,3,4,6

Explanation:

Keyword yield needed to return value. We need to add case EIGHT or default section. We need to replace return by yield. In case SIX due to new switch syntax we must change from : to ->. When switch with enum covers all cases we do not need default clause. Keword break after semicolon will cause compile error.

- https://docs.oracle.com/en/java/javase/17/language/switch-expressions.html

Date/Time API

10. Select statements that are true about Period and Duration in java.

1. Period and Duration can be used with LocalDate, LocalDatetime and date classes.
2. Period can be used with LocalDate, LocalDatetime and ZoneDateTime.
3. Duration can work with LocalDate.
4. Period.of(1,2,3) will create period with 1 year, 2 months and 3 days
5. Duration works with week, days, hours and seconds

Correct Answer: 2,4

Explanation:

1st is false - Duration not work with LocalDate because it consist of hour, seconds which is not used in LocalDate.2nd is true - Period can be used with LocalDate, LocalDateTime, ZoneDateTime. Period deals with year, month and days and all above classes has mentioned items. 3rd is false - Duration not work with LocalDate. LocalDate.now().plus(Some Duration value) will throw exception. 4th is true - it is factory method to create Period of 1 year, 2 months and 3 days. 5th is false - Duration works with days, hours, minutes and seconds but not weeks.

- https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

11. Select statements that are true about Period and Duration in java.

1. Duration deals with days, hours, seconds, millis, nanos.
2. Period.of(1) can be added to LocalTime
3. var d365 = Duration.ofDays(365);LocalDateTime.now().plus(d365); - will compile
4. LocalTime can be used with Duration only.
5. LocalDate can be used with Duration.

Correct Answer: 1,3,4

Explanation:

1st is true - Duration deals with days, hours, seconds, millis, nanos. 2nd is false - Period does not work with LocalTime because it deals with year, month and days it's more likely to use Duration. 3rd is true - it will compile but will throw runtimeException ' Unsupported unit: Seconds'. 4th is true - Duration and LocalTime deals with hours, seconds, minutes and so on. 5th is false - LocalDate deals with Period (year, month and days)

- https://docs.oracle.com/javase/tutorial/datetime/iso/index.html

Exception Handling

12. Select statements that are true about exceptions in java.

1. try with resource works with objects that implements Closeable.
2. try with resources, do not release resources implicitly.
3. try catch finally block works more effectively with classes that implements AutoClosable rather than try with resource.
4. finally block do not throw exceptions.
5. When using finally programmer will not have a resource leak.

Correct Answer: 1

Explanation:

1st is true - Java classes that implements Closable or AutoClosable are best to use with try with resource because developer does not need to close resource explicitly and resources get released after their execution. 2nd is false - try with resources handles it implicitly. 3rd is false - see 1st answer. 4th is false - Resources that are get closed in finally block may throw exception. 5th is false - Resources that are get closed in finally block may throw exception and many unclosed resources may lead to memory leak.

- ➢ https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

13. What is the result of the code below?

```
try (FileReader reader  = new FileReader("file.txt");
     var bufferedWriter = new
BufferedReader(reader);
bufferedWriter = null;) {
    throw new IllegalArgumentException();
} catch (Exception e) {
    System.out.println("Exception");
} catch (RuntimeException e) {
    System.out.println("RuntimeException");
}
```

1. Exception
2. RuntimeException
3. ExceptionRuntimeException
4. code will not compile
5. none of the above

Correct Answer: 4

Explanation:

Line with 'bufferedWriter = null' will throw such exception: the try-with-resources resource must either be a variable declaration or an expression denoting a reference to a final or effectively final variable

- ➢ https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

14. What is the result of the code below?

```
     try (var reader  = new FileReader("file.txt")) {
        throw new FileNotFoundException();
     } catch (Error e) {
        System.out.print("Error");
throw new RuntimeException();
     } catch (Exception e) {
        System.out.print("Exception");
     }
```

1. ErrorException
2. Error
3. Exception
4. code will not compile
5. none of the above

Correct Answer: 3

Explanation:

FileNotFoundException extends IOException which is child of Exceptions so catch(Exception e) will be executed.

- ➢ https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

Garbage collection

15. Select statements that are correct about garbage collection.

```
1   class GCZoo {
2     public void referenceZoo(GCZoo zoo) {
3         GCZoo localZoo = zoo;
4     }
5     public static void main(String[] args) {
6          GCZoo zooOne = new GCZoo();
7          GCZoo zooTwo = new GCZoo();
8          new GCZoo().referenceZoo(zooOne);
9          zooOne = null;
10     zooTwo = null;
11         System.gc();
12      }
13   }
```

1. object on line 6 is eligible for garbage collection after line 9
2. GCZoo object on line 8 is not eligible until the end of program
3. GCZoo object on line 8 is eligible right after execution
4. object on line 6 has additional reference on line 3 and will live until end of program
5. object on line 7 is eligible for garbage collection after line 10

Correct Answer: 1,3,5

Explanation:

1st is true - after line 9 there is no more reference to that object. 2nd is false - GCZoo object is eligible for gc right after execution. 3rd is true see answer of option 2. 4th is false - since referenceZoo has only local variables they live only in that scope so when code goes out of that scope they become eligible also. 5th is true - Object on line 7 lose its reference on line 10 and become also eligible for gc.

Java OOA instanceof

16. Select statements that are correct alternatives of code inside the runExceptions method.

```
public  void runExceptions(RuntimeException e) {
      if(e instanceof ClassCastException) {
         ClassCastException exception =
(ClassCastException) e;
         System.out.println(e.getMessage());
      }
}
```

1.
```
if(e instanceof ClassCastException cce) {
   System.out.println(cce.getMessage());
}
```

2.
```
if(e instanceof final InterruptedException cce) {
   System.out.println(cce.getMessage());
}
```

3.
```
if(e instanceof final ClassCastException cce ||
cce.getMessage() != null) {
   System.out.println(cce.getMessage());
}
```

4.
```
if(e instanceof final ClassCastException cce) {
   cce = (ClassCastException) e;
   System.out.println(cce.getMessage());
}
```

Correct Answer: 1

Explanation:

1st is valid - valid syntax of instance of. 2nd is false - it used InterruptedException instead of ClassCastException. 3rd is false - it uses || operation which can lead to NullPointerException when instanceof condition will be false. 4th is false they are trying to reassign final variable.

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-instanceof-operator.html

Java OOA Overloading, Overriding, Interfaces

17. Select statements that are true about inheritance in java(Child class inherits Parent class).

1. A child class inherits public and protected members of its parent.
2. A child class does not inherit the package-private members of the parent that is located in same package.
3. Private members of its parent class can not be accessed even through public getter methods.
4. Child child = new Parent() will compile.
5. Parent child = new Child() will compile.

Correct Answer: 1,5

Explanation:

1st is true, 2nd is false - because child class in same package has access to default members of parent. 3rd is false - private members are accessible through public getter methods. 4th is false - will throw exception that Parent cannot be converted to Child class. 5th is true - valid syntax.

- ➢ https://docs.oracle.com/javase/tutorial/java/concepts/index.html

18. What will be the output of the code when Rhino and Zebra class are executed?

```
package javaOOA.one;
public class Animal {
    String className = "Animal";
    protected String type = "Wild";
}
¦
package javaOOA.one;
public class Rhino extends Animal{
    public Rhino() {
        System.out.print(this.className+""+this.type);
    }
    public static void main(String[] args) {
        new Rhino();
    }
}
¦
package javaOOA.two;
import javaOOA.one.Animal;
import javaOOA.one.Rhino;
public class Zebra extends Animal {
    public Zebra() {
        System.out.print(this.className);
        System.out.print(this.type);
    }
    public static void main(String[] args) {
        new Zebra();
    }
}
```

1. WildAnimal, will not compile
2. AnimalWild, will not compile
3. AnimalWild, AnimalWild
4. Both variants will not compile
5. AnimalWild, nullWild

Correct Answer: 2

Explanation:

Since Rhino is in same package it has access to instance fields of Animal but Zebra is located in another package and do not have access to package-private field className.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

19. Choose correct statements about access modifiers.

1. Protected fields are accessible in subclasses that are in the same package.
2. Protected fields are accessible in subclasses in another package.
3. Protected fields are accessible at random classes that are in the same package.
4. fields without modifiers are accessible from subclasses in another package.
5. none of the above

Correct Answer: 1,2,3

Explanation:

Protected fields are accessible in child class in same and outer packages. They are also accessible when random class in same package creates instance of Class with protected fields and calls one of them. 4th and 5th are false - Field with default modifier is not accessible in subclass of another package.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

20. What will be the result when we run

```
BaseChildOne, BaseChildTwo, NoParent?
package javaOOA.one;
public class Base {
   String defaultHello = "Default";
   protected String protectedHello = "Protected";
}
...
package javaOOA.one;
public class BaseChildOne extends Base{
   public BaseChildOne() {
System.out.print(this.defaultHello + "" +
this.protectedHello); }
}
...
package javaOOA.two;
import javaOOA.one.Base;
public class BaseChildTwo extends Base {
   public BaseChildTwo() {
System.out.println(this.defaultHello + "" +
this.protectedHello);}
}
...
package javaOOA.two;
import javaOOA.one.Base;
public class NoParent {
   public NoParent() {
      Base base = new Base();
      System.out.println(base.protectedHello);}
}
```

1. DefaultProtected, will not compile, will not compile
2. DefaultProtected, DefaultProtected, will not compile
3. DefaultProtected, will not compile, Protected
4. DefaultProtected, nullProtected, will not compile

Correct Answer: 1

Explanation:

Package private field defaultHello is accessible in child class in same package only. Thats why only BaseChildOne will be executed others will not compile.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

21. Choose correct statements about inheritance.

1. super keyword - is used to call parent constructor
2. keyword 'this'- is used to reference instance and parent instance fields.
3. super() - is used to call parent constructor
4. A child class in the same package inherits only public and protected members of its parent.
5. A child class of record class can access public and protected members.

Correct Answer: 2,3

Explanation:

keyword super is used to reference parent fields. 2nd is true. 3rd is true. 4th is false - child class will own public, protected and default members of parent in same package. Record class cannot be inherited because it is final class.

- https://docs.oracle.com/javase/tutorial/java/concepts/index.html

JDBC

22. Select statements that are true about PreparedStatement methods.

1. executeUpdate() can update, delete, select, insert rows
2. execute() can be used to update rows.
3. execute(),executeUpdate() can update rows.
4. execute() can update, delete, select, insert rows
5. executeQuery() used to select rows.
6. none of the above

Correct Answer: 2,3,4,5

Explanation:

1st is false - executeUpdate - executes the SQL statement in this PreparedStatement object, which must be an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE. 2nd, 3rd, 4th, 5th are true.

➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

23. Fill in the blanks with correct statements for table demo with columns id of type bigint and name of type text.

```
var sql = "INSERT INTO demo(name, id) ________";
```

```
try (Connection conn =
DriverManager.getConnection(url)) {
   PreparedStatement ps =
conn.prepareStatement(sql);
   ________
   ps.executeUpdate();
}
```

1. 1 - VALUES(?, ?)
   2 - ps.setLong(1, 1); ps.setString(2, "Arthur");
2. 1 - VALUES(?, ?)
   2 - ps.setString(0, "Arthur");ps.setLong(1, 1);
3. 1 - VALUES(?, ?)
   2 - ps.setString(1, "Arthur");ps.setLong(2, 1);
4. 1 - VALUE(?, ?)
   2 - ps.setLong(1, 1); ps.setString(2, "Arthur");
5. 1 - VALUES({}, {})
   2 - ps.setString(1, "Arthur");ps.setLong(2, 1);

Correct Answer: 3

Explanation:

1st is false - because first column must be string value.2nd is false - parameter index starts from 1. 3rd is true - correct usage. 4th is false - keyword VALUE is invalid. 5th is false - incorrect placeholders {} must be ?

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Lambda and Functional Interface

24. Select statements that are true about lambda and functional interface.

1. Interface with methods same as in Object class will not be a valid functional interface.
2. Interface with method to toString with return type String and with parameter is valid functional interface.
3. System.out::println works the same as lambda s -> System.out.println(s).
4. Functional interface can not have a default method.

Correct Answer: 1,2,3

Explanation:

Answer for option 1, 2, 4 - Conceptually, a functional interface has exactly one abstract method. Since default methods have an implementation, they are not abstract. If an interface declares an abstract method overriding one of the public methods of java.lang.Object, that also does not count toward the interface's abstract method. 3rd option also valid

- https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

Localization

25. What will be the output of the code below?

```
Locale.setDefault(Locale.ENGLISH);
var bundle = ResourceBundle.getBundle("i10n",
Locale.FRENCH);
System.out.print(bundle.getString("greeting"));
System.out.print(bundle.getString("bye"));
bundle = ResourceBundle.getBundle("i10n");
System.out.print(bundle.getString("greeting"));

Our resources folder contains 3 files with values.
i10n.properties
greeting=GoodMorning
bye=GoodBye

i10n_en.properties
greeting=Hi

i10n_fr.properties
greeting=Bonjour
```

1. BonjourHi
2. BonjourNullHi
3. BonjourGoodByeHi
4. GoodMorningGoodBye
5. HiGoodByeBonjour
6. BonjourHiGoodBye

Correct Answer: 3

Explanation:

value of 'greeting' will be 'Bonjour' because we set locale to French explicitly. Value of 'bye' will be 'Goodbye' because neither of i10n_fr, i10n_en contains key 'bye'. And result of second 'greeing' will be 'Hi' because it uses default Locale which is English. Order of properties how java will get localiations is i10n_fr then default locale which is english i10n_en then i10n file.

- ➢ https://docs.oracle.com/javase/tutorial/i18n/

26. Select statements that are true about Localization in java.

1. ResourceBundle works with property files that contain localizations.
2. when a resource bundle does not contain a key then it will fail silently without throwing error.
3. The default resource bundle will be checked for needed resources firstly.
4. Resources will be taken from the default resource bundle in a latest order if other files do not contain it.

Correct Answer: 1,4

Explanation:

1st is true, 2nd is false - when key not found java will throw exception. 3rd is false - default resource bundle is seen last by order. 4th is true - firstly will look for matched resource bundle then default bundle.

- ➢ https://docs.oracle.com/javase/tutorial/i18n/

Math API

27. What will be the output of the code below?

```
var decimal  =  new BigDecimal("12122022");
var integer  =  new BigInteger("12122022");
Number number  =  new BigDecimal("12122022");
System.out.print(decimal.equals(integer)+""+decim
al.equals(number)+decimal.compareTo((BigDeci
mal) number));
```

1. truetrue0
2. falsefalse-1
3. falsetrue0
4. code will not compile
5. truetrue1
6. none of the above

Correct Answer: 3

Explanation:

equals() - compares this BigDecimal with the specified Object for equality. Unlike compareTo, this method considers two BigDecimal objects equal only if they are equal in value and scale. In current example decimal.equals(number) will be true, but decimal.equals(integer) will be false because integer does not have scale. Third operation compareTo will return 0. Scale is the number of digits to the right of the decimal point.

- https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Modules

28. Select statements that will help you to compile and run the utils module with the class Main within package com.sample.

1. java --module-path out -m utils/com.sample.Main
2. javac --module-source-path <folder name containing utils> -m utils -d out
3. java --module-path out -m com.sample.Main
4. javac --module-source-path src/main/java -m com.sample.Main -d out
5. javac --module-source-path src/main/java -m utils/com.sample.Main -d out
6. java --module-source-path -m utils/com.sample.Main -d out

Correct Answer: 1,2

Explanation:

option 1,2 will compile and run. Others will not work

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

29. Consider a case when there are 2 modules (m.base, m.helper) where m.base is using a

method from the m.helper. Select statement(s) that helps to do that .

1. add exports <package>; in m.helper module-info.java; add requires m.helper; in m.base module-info.java
2. add exports m.helper; in m.helper module-info.java; add requires m.helper; in m.base module-info.java
3. add exports <package>; in m.base module-info.java; add requires m.helper; in m.helper module-info.java
4. don™t need to write explicitly requires and exports when modules in the same project directory
5. as java.base there is no need to write requires m.helper explicitly in module-info.java
6. none of the above

Correct Answer: 1

Explanation:

we need to add 'export <package of m.helper >' in m.helper module-info.java and add 'requires m.helper' in m.base module-info.java. 2nd is false - exports <module> is not correct. 3rd is false - we need to add 'exports <package of m.helper >' in m.helper module. 4th is false - we need to write explicitly export and requires. 5th is false - we need to export of package in m.helper and requires it in m.base.

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

30. Consider the case when you need to create a GUI app using java. Select statement(s) that help to do it.

1. add module-info.java into working module; add exports java.desktop;
2. add module-info.java into working module; add requires javax.swing;
3. add module-info.java into working module; add requires java.desktop;
4. since javax.swing is core java api no need to write require explicitly
5. add module-info.java into working module; add requires java.se;
6. none of the above

Correct Answer: 3,5

Explanation:

GUI packages are located in java.desktop module. Also we can require java.se module which combine java.desktop also. We need to add into our module-info.java requires java.desktop; or requires java.se;

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

31. Select statements that are true about modules.

1. require modules other than java.base is not mandatory in java app
2. require java.base will not cause compile error
3. require one module multiple times will case compile error
4. requires can be transitive and static
5. it is compile-time error if there are two exports of package with same name

Correct Answer: 3, 4, 5

Explanation:

1st is false - only java.base module is not needed to require explicitly. 2nd is false - java.base is required implicitly, if we require it explicitly there will be compile error. 3rd is true, 4th is true, 5th is true.

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-7.html

Arithmetic/Boolean Expressions (Operators, Promotion, Casting)

32. What is the data type of variables numc, numf, numg?

```
int numa = 10;
long numb = 5;
var numc = numa + numb;
double numd = 2.0;
float nume = 4.0f;
var numf = numd + nume;
var numg = (short)10 * (int)5;
```

1. int, float, short
2. int, double, int
3. long, double,short
4. long, double, int
5. long, float, short
6. none of the above

Correct Answer: 4

Explanation:

long + int = long; double + float = double; short + short = integer. When math operation occurs between different types then value is converted to greater type.

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

33. What is the result of the following code snippet?

```
int a = 31;
double b = a%7/2;
double result = ++a + b;
System.out.println(result);
```

1. 32.5
2. 31.5
3. 33.0
4. 32.0
5. 31.0
6. none of the above

Correct Answer: 3

Explanation:

1. a%7 is 3   2. b = 3/2 = 1.0   3. ++a = 32; 32 + 1.0 = 33.0

- ➢ https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

34. What is the result of the following code snippet and type of numD ?

```
int numA = 20;
double numC = numA/3;
var numB = (numA++) + numC;
var numD = 20L + ++numB;
System.out.println(numD);
```

1. 48.0; double;
2. 48; long
3. code will not work
4. 47.0; double
5. 46.0; double
6. 46; long

Correct Answer: 4

Explanation:

numc equals 6.0 because when we divide one integer on another integer result will be also integer. numB equals 26.0 because postincrement will return value which is 20 and then increment so 20 + 6.0 = 26.0. numD equals to 47.0 because 20L + 27.0 (preincrement will increase and then return value) will give us double 47.0. When math operation occurs between different types then value is converted to greater type. Ex: int + double = double, long + double = double

- https://docs.oracle.com/javase/specs/jls/se17/html/jls-5.html

Record

35. Select variants of correct usage of record class.

1. public class record RecordClass() {}
2. public record RecordClass() {}
3. public record RecordClass {}
4. public record RecordClass(String title) {}
5. public record RecordClass(String title) {
   public RecordClass { System.out.println(title); }}
6. public record RecordClass(String title) {
   public RecordClass(String title) {
   System.out.println(title); }}

Correct Answer: 2,4,5

Explanation:

1st is invalid - keyword class is extra. 2nd is valid, 3rd is invalid - we need to add parentheses after recordName. 4th is valid. 5th is valid. 6th is invalid - we need to add this.title = title in body of constructor otherwise will compile error occur.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

36. Select variants of correct usage of record class.

1. public record RecordSample(int value) {
     public void value() { System.out.println(value);}}

2. public record RecordSample(int value) {
     public int value() { return value;}}

3. public record RecordSample() {
     static String title;}
   public record RecordSample() {
     public final String title;}

4. public record RecordSample(int value) {
     public RecordSample {
   System.out.println(value);}}

5. none of the above

Correct Answer: 2,3,5

Explanation:

1st is false - record's value() method must return 'value' variable by record definition.2nd is true - see answer 1. 3rd is true - Record can have static instance fields. 4th is false - instance fields must be static. 5th is true - record can have a Constructor.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

37. Select statements that are correct usage of record.

1. record Zoo<T extends Animal> (T mammal, T predator) { }

2. record Zoo {}

3. record Zoo() implements Animal {}

4. public record Zoo() {
       record WhaterAnimal() {}}

5. public record Zoo() {
       public Zoo0 {}}

6. record Zoo<T extends Animal> (T mammal, T predator) {
       public final String name;
     }

Correct Answer: 1,3,4

Explanation:

1st is valid, 2nd option needs parantheses. 3rd is valid - record can implement interface. 4th is true - record can have nested record. 5th is false - record constructor name is different. 6th is false - record instance fields must be static.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

Sealed Classes

38. Select statements that are true about sealed classes.

1. sealed class and its child classes can locate in different packages
2. sealed class can implement sealed interface only
3. sealed class cannot have interface
4. developer can identify whether class is sealed by calling isSealed() method from java.lang.Class
5. sealed class and its child classes must locate in same package
6. none of the above

Correct Answer: 4,5

Explanation:

1st is false - they must be in same package. 2nd is false - it can implement ordinary interface. 3rd is false - sealed class can implement interface. 4th is true, 5th is true see answer 1.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- https://openjdk.org/jeps/412

39. Select statements that are true about sealed classes.

1. identify whether class is sealed by calling isSealed() method from java.lang.Class
2. sealed class must always use permits keyword
3. sealed class may omit permits keyword when child classes are in same file
4. child classes with final, non-sealed modifiers can not have its own child classes
5. sealed class must use permits when child classes are in another package and omit permits when child classes in same package
6. none of the above

Correct Answer: 1,3

Explanation:

1st is true, 2nd is false - if parent and children classes(nested) are in the same java class then parent can omit keyword 'permits'. 3rd is true - see answer 2. 4th is false - Non-sealed child classes can have its own children classes. 5th is false - In sealed inheritance parent and child class must be in same package.

- ➢ https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- ➢ https://openjdk.org/jeps/412

40. Select statements that are true about sealed classes.

1. child classes may or may not have modifiers final, sealed,non-sealed
2. interface that extend sealed interface must have modifier final or sealed or non-sealed
3. sealed interface cannot have child final interface
4. child classes with final, non-sealed modifiers can not have its own child classes
5. none of the above

Correct Answer: 3

Explanation:

1st- Sealed class children must be final, sealed or non-sealed. 2nd- interface can't be final (throw exception: illegal combination of modifiers â˜’ interface and final) . 3rd - is valid see answer of option 2. 4th - Non-sealed child classes can have it's own children classes.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- https://openjdk.org/jeps/412

Streams

41. Select statements that are true about streams.

1. intermediate operations will not run without terminal operation
2. intermediate operations can run without terminal operation
3. flatMap(list -> list.stream()) - correct usage of flatMap() method
4. map() is used to create map from list
5. var stream = Stream.of(1,2,3); for(Integer i : stream) { } - will compile
6. stream can have multiple terminal operations

Correct Answer: 1,3

Explanation:

1st is true - streams use lazy evaluations, so if we do not add terminal operation, stream will not executed. 2nd is false - see answer 1. 3rd is valid. 4th is false - map returns a stream consisting of the results of applying the given function to the elements of this stream. Stream can"t be used with enhanced loop. 6th is false - only single terminal operation is allowed.

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

42. Select correct variants of min function.

```
Comparator<Integer> comparator = (x,y) -> (x < y)
? -1 : ((x == y) ? 0 : 1);
var list = List.of(-9, 1, 3, 4);
```

1. list.stream().min()
2. list.stream().mapToInt(v -> v).min()
3. list.stream().min(Integer.compare(x,y))
4. list.stream().min(Integer::compare)
5. list.stream().min(comparator)
6. none of them

Correct Answer: 2,4,5

Explanation:

1st is invalid - min() function of stream needs comparator as a parameter. 2nd is valid - stream converted to IntStream and then called min method. 3rd is invalid -min(Integer.compare(x,y)) - syntax error. 4th and 5th options are valid.

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

43. Select correct variants of max function.

```
Comparator<Integer> comparator = (x,y) -> (x < y)
? -1 : ((x != y) ? 0 : 1);
var list = List.of(9, 21, 3, 24);
```

1. list.stream().mapToInt(v -> v).max()
2. list.stream().max()
3. list.stream().max(Integer::compare)
4. list.stream().max(Integer.compare(x,y))
5. list.stream().max(comparator)

Correct Answer: 1,3

Explanation:

1st is valid - when we use mapToInt our stream will be converted to IntStream which has method max(). 2nd is invalid - max() method needs comparator. 3rd is valid - max() method with comparator. 4th is invalid - max(Integer.compare(x,y)) will cause syntax error. 5th is invalid - comparator given in example is incorrect so option 5 is invalid.

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

44. What will be the correct output of code below?

```
var nums = Stream.of(1,3,4,51,4,5,9,21);
nums.distinct().limit(6).peek(System.out::print).collect(Collectors.toList());
```

1. 124514
2. 134591

3. 134592
4. 421233
5. 13459
6. 1345159

Correct Answer: 6

Explanation:

after distinct() operation we will have [1, 3,4 ,51,5,9,21]. After limit(6) stream will contain [1,3,4,51,5,9] so output will be 1345159.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

45. What will be the correct output of code below?

var nums = Stream.of(5,6,2,6,21,8,9,1,4);

nums.distinct().limit(5).reduce((l,k) -> l+k).min();

1. sum of all distinct numbers
2. sum of all distinct numbers that are first 5 in stream
3. code will not compile
4. 42
5. 41

Correct Answer: 3

Explanation:

min() - method is called on stream, but reduce() will return Optional. Code will not compile

- ➢ https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

Text Block

46. What will be the output of the code below?

```
String a = "",b = "";
var sb1 = new StringBuffer(), sb2 = new
StringBuffer();
System.out.println(a.equals(b) + "-" +
sb1.equals(sb2) + "-" + a.equals(sb2));
```

1. code will not compile
2. true-false-false
3. true-true-false
4. true-true-true
5. false-true-false
6. true-false-true

Correct Answer: 1

Explanation:

var' is not allowed in a compound declaration - so code will not compile

- ➢ https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

47. Select statements that are true about the String, StringBuffer, StringBuilder.

1. immutable, mutable, mutable
2. immutable, mutable, immutable

3. immutable, immutable, immutable
4. mutable, immutable, mutable

Correct Answer: 1

Explanation:

String is immutable and all operations with string will create new string value. StringBuffer and StringBuilder are mutable, we can add or replace its items and will work only with single object.

- https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

48. What will be the output of the code below?

```
      var a = new StringBuilder(); var b = new
StringBuilder();
      var c = new StringBuilder(); var d = new
StringBuilder();
      d = c.append("lorem"); d = d.append("ipsum");
          System.out.println("" + a.equals(b) +
d.equals(c) + c.equals(a.append("lorem")));
```

1. falsefalsefalse
2. falsetruetrue
3. falsefalsetrue
4. falsetruefalse
5. truetruetrue
6. truefalsefalse

Correct Answer: 4

Explanation:

d and c are references same StringBuilder so their comparison will return true. a and b are different objects their comparison will return false. c and a are also different objects with different values.

- https://docs.oracle.com/en/java/javase/15/text-blocks/index.html

Local Variable Type Inference

49. Select statement(s) that are true about "var" type?

1. var type is inferred from context
2. var type is inferred on runtime;
3. var type can be used outside method
4. type of var can be changed on runtime
5. var type can be declared without initialization

Correct Answer: 1

Explanation:

When var type is declared and initialized it will infer type from context. 2nd is false - it is inferred at compile time. 3rd is false - it is used inside methods, constructors and not as instance fields. 4th is false - Var type does not change after being initialized. 5th is false - after declaration it must be initialized.

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

50. Look at the code below and select statements that are correct for local variable inference.

```
4. var a = "Hello";
5. var c=null;
6. a=null;
7. var b = 1; b = "World";
8. System.out.println(a+"  "+b);
```

1. will print "Hello World";
2. will not compile due to row 5.
3. will not compile due to row 7.
4. will not compile due to row 6.
5. none of the above

Correct Answer: 2,3

Explanation:

var type - can't be initialized with null value so line 5 will not compile. var type - can't be changed from int to string so line 7 will not compile. Line 6 is ok because var a was correctly initialized and then reassigned with null value.

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

Streams

1. Select statements that are correct structure of stream pipeline.

1. intermediate operation(s) - terminal operation
2. source - intermediate operation(s) - terminal operation
3. multiple sources - terminal operations
4. source - intermediate operation

Correct Answer: 2

Explanation:

Stream pipeline consist of - source, intermediate operations, terminal operation.

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html

2. Consider 2 cases and select best option.

```
//case 1
 List vs = List.of("v1", "v2", "v3");
 vs.stream().peek(i -> {
    System.out.print(i);
    }).collect(Collectors.toList());

    //case 2
    List vs = List.of("v1", "v2", "v3");
    vs.stream().forEach(i -> {
       System.out.print(i);
    }).collect(Collectors.toList());
```

1. case 1 will print v1v2v3
   case 2 will print v1v2v3

2. case 1 will not compile
   case 2 will print v1v2v3

3. case 1 will print v1v2v3
   case 2 will not compile

4. none of them

Correct Answer: 3

Explanation:

case1 - will work because peek() is intermediate operation and collect is terminal.

case2 - forEach() is terminal operation and and has return type void so calling another terminal operation collect() will cause compile error.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html

3. Consider a List<ClassRoom> classrooms where ClassRoom has field List<String> studentNames. Try to create total list of student names without duplicates in lowercase but starting with capital letter.

1. classrooms.stream().distinct().map(i -> i.studentNames.stream()).distinct().map(name -> name.substring(0, 1).toUpperCase() + name.substring(1).toLowerCase()).collect(Collectors.toList());

2. classrooms.stream().distinct().map(name -> name.substring(0, 1).toUpperCase() + name.substring(1).toLowerCase()).collect(Collectors.toList());

3. classrooms.stream().flatMap(i -> i.studentNames.stream()).peek(i -> { i.distinct()}).map(name -> name.substring(0, 1).toUpperCase() + name.substring(1).toLowerCase()).collect(Collectors.toList());

4. classrooms.stream().flatMap(i -> i.studentNames.stream()).map(name -> name.toUpperCase()).collect(Collectors.toList());

5. classrooms.stream().flatMap(i -> i.studentNames.stream()).distinct().map(name -> name.substring(0, 1).toUpperCase() + name.substring(1).toLowerCase()).collect(Collectors.toList());

Correct Answer: 5

Explanation:

option 1 and 2 are not using flatMap(). Option 3 using peek() function incorrectly. Option 4 - not using distinct() and converts whole name to uppercase. option 5 is correct

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html

4. What will be result of the code below?

```
        Stream<String> stream = Stream.of("h", "i", "1", "3");
         int len = stream.reduce(0, (a,b) -> a +
b.length(), (x, y) -> x + y);
        int len2 = stream.reduce("", (a,b) -> a + b).length();
         System.out.print(len);
         System.out.print(len2);
```

1. 22
2. 4
3. throws java.lang.IllegalStateException
4. 44

Correct Answer: 3

Explanation:

Single stream used twice in reduce() function. Stream after calling terminal operation can't be used again, we just need to create new one.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html

5. Select all statements that will return result from stream.

1. Stream<Double> stream = Stream.generate(() -> Math.random()*10);
2. multiple-choice
3. stream.mapToDouble(i -> i).max().getAsDouble()
4. stream.mapToDouble(i -> i).reduce(0, (a,b) -> a + b)
5. stream.mapToDouble(i -> i).min().getAsDouble()
6. stream.mapToDouble(i -> i).findAny().getAsDouble()
7. stream.mapToDouble(i -> i).average()

Correct Answer: 4

Explanation:

Only findAny() method will return results from this infinite stream. Others will wait until it finishes execution which will never occur.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/stream/Stream.html

Var – local variable inference

6. What will be output of the code below if we call new Q4Samples()?

```
3  public class Q4Samples {
4  public int count;
5  public Q4Samples() {
6     var count = this.count;
7     var localCount = 11;
8     localCount = null;
9  }}
```

1. will not compile, because of row 6
2. will not compile, because of row 8
3. will not compile, because of row 11
4. will compile
5. both rows 6, 8 will cause compile error

Correct Answer: 2

Explanation:

Field count on line 4 has default value so line 6 will be valid, line 8 is not valid because we are trying to assing null on primitive.

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

7. What we need to modify here to make this method valid?

```
2   public void var(var a) {
3      String var;
4      if(a == true) {
5         var = "is true";
6      }
9      var localVar = var;
10   }
```

1. fix line 3, change variable name

2. fix line 2: change type of variable 'a' to boolean

3. add else { var = "is false"; } statement

4. will compile

Correct Answer: 2, 3

Explanation:

1st - line 3 is valid, so no need to fix it. 2nd - We need to change var a to boolean a, we cannot use var as method parameter. 3rd - We need to add else { var = "is false"; } because if there will be case when a = false then var will not be initialized and will cause compile error .4th - Without modifications code will not work.

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

Arithmetic/Boolean Expressions(Operators, Promotion, Casting)

8. What will be the output of the code below?

```
int b = 10, c = -1;
int a = ++b+c++;
System.out.print(""+a+b+c);
```

1. will not compile
2. 22
3. 23
4. 11110
5. 10110
6. 10111

Correct Answer: 5

Explanation:

increments(++) will be executed firstly. ++b will increment and return value so b = 11, c++ will return value and then increment so at moment of addition c = -1. a = 11 + (-1) = 10. On output we added empty string to int so whole result will be just string concatenation. Output = "" + a + b + c = 10110. (c is 0 because of postincrement operation)

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

9. What will be the output of the code below?

```
int x = 11, z = 2, w = 15;
 float f = 10.5f;
 f += f++;
 var result = z*(x++)-(--w)+f;
 System.out.println(result);
```

1. 31
2. 30
3. 28
4. 29
5. 30

Correct Answer: 4

Explanation:

f +=f++ operation equals f = f + f ++;

f ++ will return value and then increment.

f = 10.5 + 10.5 = 21.0 because = has lowest order precedence it will overwrite value of f which will be 11.5 after postincrement.

var result = 2 * 11(will return value then increment) - (decrease value and return it)14 + 21.0 = 29.0

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

String/Text

10. What will be the output of the code below?

```
 String a = "java", b = "j";
 String c = "jdk", d ="";
 b = (b + "ava").intern(); d +="jdk";
 System.out.println("" + (a==b) + (d==c) +
d.equals(c));
```

1. falsefalsetrue
2. truetruetrue
3. truefalsetrue
4. falsefalsetrue
5. none of them

Correct Answer: 3

Explanation:

a and b will be equal because both of them are referencing value from string pool. d == c will be false because d does no referencing value in string pool so different references will result to false. d.equals(c) will result true because it will compare actual values.

- ➢ https://docs.oracle.com/javase/tutorial/java/data/strings.html

11. What will be the output of the code below?

```
var sb = new StringBuilder("ocp");
var sb2 = sb.append(" java");
var sb3 = new StringBuilder("ocp java");
System.out.println("" + (sb == sb2) +
(sb2.equals(sb3)) + (sb2 == sb3));
```

1. truefalsefalse
2. truetruefalse
3. truefalsetrue
4. truetruetrue
5. none of them

Correct Answer: 1

Explanation:

sb == sb2 will return true because they are references same object.

sb2.equals(sb3) will return false equals() method compare their references that differs. sb2 == sb3 will return false because it's also compares object references

- https://docs.oracle.com/javase/tutorial/java/data/strings.html

12. What will be the output of the code below?

```
String block = """
       block 1""".trim();
var liter = "block";
```

```
int one = 1;
System.out.print(("block" + " 1") == block);
System.out.print(("block " + one) == liter);
```

1. falsetrue
2. falsefalse
3. truetrue
4. truefalse
5. none of them

Correct Answer: 4

Explanation:

"block" + " 1" will be evaluated at compile time that's why it will be stored a string pool and will return true when we compare it with variable block.

"block " + one will be evaluated at runtime because string concatenates with variable and new string created that is not in string pool so when we compare with variable block returns false.

- ➢ https://docs.oracle.com/javase/tutorial/java/data/strings.html

Sealed Classes

13. What is the output of the code below?

```
public sealed interface SealedInterface permits
SealedParent{
   void getModifier();
   default void parentModifier()
{System.out.println("sealed");}}
....
public sealed class SealedParent implements
SealedInterface permits Child{
   @Override
   public void getModifier() {
      System.out.println("sealed");
      this.parentModifier();}}
....
public final class Child extends SealedParent {
   @Override
   public void getModifier() {
      System.out.println("final");}}
....
main method

SealedParent sp = new Child();
Child ch = new Child();
sp.getModifier();
ch.getModifier();
```

1. code will not compile, interface can't be sealed

2. final
   final

3. sealed
   sealed
   final

4. code will not compile, because child of sealed class that implements interface can't be final

5. none of them

Correct Answer: 2

Explanation:

code will compile. Syntax is valid.
SealedParent sp = new Child() - will create instance of Child
Child ch = new Child() - - will create instance of Child
sp.getModifier() - so output will be final
ch.getModifier() - output will be final

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html

14. ___ modifier allows other classes to extend the current class without permission, even if it's a subclass of a sealed class.

1. public

2. final

3. sealed

4. non-sealed

5. protected

Correct Answer: 4

Explanation:

Only non-sealed class can be extended with random class without permission.

- ➢ https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html

15. Select statements that are true about sealed classes.

1. a sealed class can be extended with any class
2. sealed class must be implemented
3. keyword 'permits' may not be used when children of sealed class are located in the same file
4. keywordsealed™ must come before the keywordclass™ when the class is declared. Ex: public sealed class Hello {}

Correct Answer: 2,3,4

Explanation:

Rules of Sealed class: it must be implemented by only permitted child classes. If children are located in the same file keywordpermits™ is optional. The keywordsealed™ must come beforeclass™ on the class declaration.

- ➢ https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html

Records

16. When the developer creates a new record class with the field of type string, there are some extra things that are created by the compiler, select all that apply.

1. Constructor
2. setter method
3. accessor method
4. Overrides equals(), hashCode(), toString() methods
5. none of them

Correct Answer: 1, 3, 4

Explanation:

The compiler creates Constructor, accessor method, equals(), hashCode(), and toString(). Records do not have setter methods.

- https://docs.oracle.com/en/java/javase/17/language/records.html

17. Consider the code when the record class has a Constructor with the same parameters. Select a statement that is true when there is such a case.

1. record class can't have a Constructor
2. the record can have a Constructor without parameters only.
3. «code will work even with an empty body of Constructor with parameters
4. need to declare assignments for all record fields.
5. none of them

Correct Answer: 4

Explanation:

Record class can have a Constructor with or without parameters. When Constructor has parameters then the developer must implicitly declare the assignment of all record fields.

- https://docs.oracle.com/en/java/javase/17/language/records.html

Math api

18. What is the output of the code below?

```
BigDecimal up = new
BigDecimal("2020.4").setScale(0,
RoundingMode.HALF_UP);
BigDecimal even = new
BigDecimal("2020.5").setScale(0,
RoundingMode.HALF_EVEN);
BigDecimal down = new
BigDecimal("2020.6").setScale(0,
RoundingMode.HALF_DOWN);
System.out.print(up + "-" + even + "-" + down);
```

1. 2020-2020-2021
2. 2020-2021-2021
3. 2021-2021-2020
4. 2021-2020-2021

Correct Answer: 1

Explanation:

HALF_UP - Behaves as for RoundingMode.UP if the discarded fraction is 0.5; otherwise, behaves as for RoundingMode.DOWN

HALF_EVEN - Rounding mode to round towards the "nearest neighbor" unless both neighbors are equidistant, in which case, round towards the even neighbor. Behaves as for RoundingMode.HALF_UP if the digit to the left of the discarded fraction is odd; behaves as for RoundingMode.HALF_DOWN if it's even.

HALF_DOWN - Rounding mode to round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down. Behaves as for RoundingMode.UP if the discarded fraction is > 0.5; otherwise, behaves as for RoundingMode.DOWN

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/math/BigDecimal.html

Localization

19. Consider case when there are exists resource bundles with same list of words. Which bundle will be used as a source?

```
Locale.setDefault(new Locale("en", "US"));
ResourceBundle bundle =
ResourceBundle.getBundle("localization");
var car = bundle.getString("car");
```

1. localization.properties
2. localization_en.properties
3. localization_us.properties
4. localization_us_en.properties
5. code will not compile

Correct Answer: 2

Explanation:

Order of resource files will be from high to low: localization_en_US.properties, localization_en.properties, localization.properties. Since there is no file localization_en_US.properties it will look into localization_en.properties. Other options are not considered. They are properties for ™us™ language code.

- ➢ https://www.oracle.com/java/technologies/localization.html
- ➢ https://docs.oracle.com/javase/tutorial/i18n/

20. What will be output of the code below?

```
l10n.properties
cartype=sedan
carmodel=Volkswagen
l10n_en_US.properties
cartype=crossover
carweight=22000kg
l10n_fr.properties
carmodel=BMW
l10n_fr_FR.properties
carweight=2500
---main
Locale.setDefault(new Locale("en", "US"));
ResourceBundle bundle =
ResourceBundle.getBundle("l10n", new Locale("fr",
"FR"));
var ct = bundle.getString("cartype");
var cm = bundle.getString("carmodel");
var cw = bundle.getString("carweight");
System.out.println(ct + "-" + cm + "-" + cw);
```

1. sedan-Volkswagen-2500

2. crossover-BMW-2500

3. crossover-Volkswagen-2500

4. sedan-BMW-2500

5. sedan-BMW-22000kg

Correct Answer: 4

Explanation:

Java will look into exactly matched resource bundles first which is fr_FR™, if not finds the key goes to fr™ if there is no such key goes to default l10n.properties file. Default locale is ignored.

- https://www.oracle.com/java/technologies/localization.html
- https://docs.oracle.com/javase/tutorial/i18n/

Exception Handling

21. Select statement(s) that will make this code work if we put it in the blank.

```
public void catchException() throws Error {
   __________
}
```

1. throw new Exception()
2. throw new RuntimeException()
3. throw new IllegalAccessError()
4. try {} catch {}
5. System.out.println("error");

Correct Answer: 2,3,5

Explanation:

Exception() - can't be handled with Errror, code will not compile.

RuntimeException and IllegalAccessError do not require to be handled because they are unchecked, so code will work. try {} catch {} - gives syntax error we need parantheses with some Exception after keyword catch. option 5 is correct. Method exception handle signature does not require method to throw exceptions.

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Exception.html

22. What is the output of the code below?

```
int array []  = new int [5];
try {
   System.out.println(array[5]);
} catch (IllegalArgumentException e) {
   System.out.print("error");
} catch (RuntimeException e) {
   System.out.print("runtime");
} catch (Exception e) {
   System.out.print("exception");
} finally {
   System.out.print("finally");
}
```

1. runtimefinally
2. finallyruntime
3. errorfinally
4. exceptionfinally
5. finallyexception

Correct Answer: 1

Explanation:

java.lang.ArrayIndexOutOfBoundsException - is thrown and it is a subclass of RuntimeException. Result will be runtime and finally at the end.

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Exception.html

23. What is the output of the code below?

```
public void throwOneMore() throw
FileNotFoundException {
   int div = 0;
   try {
      int result = 10/div;
   } catch (RuntimeException e) {
      System.out.print("div by zero");
   } catch (Exception e) {
      System.out.print("exception");
      throw new FileNotFoundException();
   } finally {
      System.out.print("0");
   }
}
```

1. div by zero0
2. code will not compile
3. div by zero
4. exception
5. exception0

Correct Answer: 2

Explanation:

code will not compile, incorrect keyword 'throw' in method exception signature. Must be 'throws'

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Exception.html

Lambda

24. Select correct variants of functional interface.

```
public interface FI {
   void run();
}
```

1.
   ```
   @java.lang.FunctionalInterface
   public interface FunctionalInterface {
      void run();
      void name();
   }
   ```

2.
   ```
   @java.lang.FunctionalInterface
   public interface Functional {
      void run();
   }
   ```

3.
   ```
   @java.lang.FunctionalInterface
   public interface FID {
      void run();
      default void name() {System.out.println("FID");}
   }
   ```
4. none of them

Correct Answer: 1,3,4

Explanation:

Functional interface - is interface with exactly one abstract method, it can contain default methods.

Annotation (@FunctionalInterface) may or may not be in interface. Option 1, 3, 4 are valid. Option 2 has 2 abstract methods.

- https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

25. Select option that will make this code call valid ((a, b) -> a * b)

1. Functional<Integer> f = (a, b) -> a * b;
2. BiFunction<Integer, Integer, Integer> biff = (a, b) -> a * b;
3. BiFunction<Integer, Integer> biff = (a, b) -> a * b;
4. Functional<Integer, Integer> f = (a, b) -> a * b;
5. UnaryOperator<Integer> ii = (a,b) -> a*b;

Correct Answer: 2

Explanation:

Option 1 is incorrect, Functional accepts 1 parameter only but we passed a and b.

Option 2 is correct. Option 3 is incorrect in BiFunction we need to provide a generic type of 2 parameters and result type. Option 4, see answer of option 1. Option 5 is incorrect, UnaryOperator can accept only one parameter but 2 provided.

- https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

26. What is the output of the code below?

```
      Predicate<Integer> pi = a -> a%2 == 0;
       Predicate<String> ps = a -> a.isEmpty();
       BiPredicate<Boolean, Boolean> bip = (a, b) -> a
&& b;
       var result = bip.test(pi.test(230), ps.test(null));
       System.out.println("re "+ result);
```

1. result-false
2. result-true
3. code will not run, because of incorrect usage of BiPredicate
4. code will not run, NullPointerException will be thrown
5. none of them

Correct Answer: 4

Explanation:

code will not run, NullPointerException will be thrown - calling isEmpty() on a null will cause exception.

- ➢ https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

Modules

27. Consider the case when you have modules a, b and c. Module a depends on packages in module b and module b is dependent on a package named thec.base™ inside module c. What line should contain module-info.java of the module c.

1. imports c.base;
2. exports c.base;
3. export c.base;
4. requires c;
5. requires c.base;

Correct Answer: 2

Explanation:

Option 1 is incorrect, we can't import package in module-info.java. Option 2 is correct. Module c should exports c.base package. Option 3 incorrect keyword 'export'. Option 4, 5 are incorrect, the keyword 'requires' is used in module that depends on another module.

- ➢ https://www.oracle.com/corporate/features/understanding-java-9-modules.html

28. Select statements that are true about modules.

1. requires - is needed to import the module.
2. requires transitive - imports the module and also makes its packages available for other modules thatrequires™ the current module.
3. exports - is needed to import the module.
4. exports - is used to export the module.
5. requires - is used to import the package from another module.

Correct Answer: 1,2

Explanation:

keyword 'requires' - is used to import module, not a package. keyword 'requires transitive' imports the module and also makes its packages available for other modules thatrequires™ the current module. Keyword 'exports' used to export package not module.

- ➢ https://www.oracle.com/corporate/features/understanding-java-9-modules.html

29. Select statements that are true about open and opens keywords.

1. opens - is used to allow all packages for the reflection mechanism.
2. open dummy.pack - is used to allow thedummy.pack™ package for the reflection mechanism.

3. opens dummy.pack - is used to allow thedummy.pack™ package for the reflection mechanism.
4. open module.name {
   opens dummy.pack
   } - will compile
5. none of them

Correct Answer: 3

Explanation:

opens <package> -

indicates that a specific package™s public types are accessible to code in other modules at runtime only. Also, all the types in the specified package are accessible via reflection.

keyword 'open' is used in declaration of module, it allows all packages inside module for reflection.

option 4 is incorrect - we can't write opens <package> inside module which has open

- keyword.https://www.oracle.com/corporate/features/understanding-java-9-modules.html

30. Select a tool that will help you to create a custom java runtime with needed modules only.

1. jdeps
2. jlink

3. jre
4. java -p <your module> jdeps build
5. jmods

Correct Answer: 2

Explanation:

Option 2 is correct.

Jlink - is a tool which helps to create your custom java runtime.

- https://www.oracle.com/corporate/features/understanding-java-9-modules.html

Java OOA instanceof

31. What is the output of the code below?

```
public static void main(String[] args) {
    isGreaterThan(5);
    isGreaterThan(11);
    isGreaterThan(20);
}
public static void isGreaterThan(Number n) {
    Predicate<Number> p = num -> num instanceof
Integer j && j > 10;
    if(p.test(n)) {
        System.out.print("greater");
    } else {
        lessThan(n);
    }
}
public static void lessThan(Number n)  {
    if(n instanceof Integer j || j < 10) {
        System.out.print("less");
    }
}
```

1. code will not compile
2. lessgreatergreater
3. lessgreater
4. greaterlessgreater
5. none of them

Correct Answer: 1

Explanation:

Code will not compile, because in the method lessThat()

line with ' if(n instanceof Integer j || j < 10)' must use && instead of ||.

There can be case when n is null and Nullpointerexception occurs.

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching.html

Java OOA GC

32. Select statements that are true about Garbage Collection in java.

1. calling System.gc() will clear eligible objects immediately
2. Objects that lost at least one of its references are eligible for garbage collection.
3. If a variable is reassigned to null its referenced Object is eligible for gc.
4. Garbage collector cleans heap and stack memory.
5. none of them

Correct Answer: 3

Explanation:

Option 1 incorrect - System.gc() wouldn't clean objects, java considers when to run this method. 2 incorrect - Objects that are lost all of its references is eligible for gc. If var a = new Integer(100); a = null; after reassigning variable a, object Integer(100) lost its references and eligible for gc. 4. GC - cleans heap not stack.

- https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html
- https://www.geeksforgeeks.org/garbage-collection-java/

33. Select statements that are true about code below.

```
public class GCollector {
   public String name;

   public void setName(String name) {
      this.name = name;
   }
   public static void main(String[] args) {
      String a = new String("a");
      String b = new String("b");
      String c = a;
      new GCollector().setName(b);
      a = null;
      b = null;
      System.out.println(c);
   }
}
```

1. String("a") - is eligible for gc right after a=null line.
2. String("b") - is eligible for gc right after b=null line.
3. String("a") and String("b") will live until end of program.
4. variable c will loose its reference after line a=null.
5. code will not compile

Correct Answer: 3

Explanation:

Both strings "a" and "b" will survive until end of program, because the String("a") has another reference that is "c" and String("b") has one more reference that is instance field "name". GC cleans objects that does not have references.

- https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html
- https://www.geeksforgeeks.org/garbage-collection-java/

JDBC

34. What is the output of the code below?

```
String insert = "insert into car values (?, ?, ?)";
PreparedStatement prep =
conn.prepareStatement(insert);
prep.setString(1, "1");
prep.setString(2, "sedan");
prep.setInt(0, 1);
prep.executeUpdate();
```

1. table car will have one more row.
2. will throw exception. Parameters are started from 1 and not 0
3. throw exception because we need to run executeQuery instead of executeUpdate().
4. table rows count will remain same because we didn't call commit() explicitly
5. none of them

Correct Answer: 2

Explanation:

Option 1 incorrect. Option 2 is correct, Parameters are started from 1 and not 0. Option 3 incorrect, we correctly called executeUpdate(), executeQuery() used to select data. Opption 4 incorrect, we don't need to call commit()

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

35. Select statements that are true about JDBC in java.

A. Statement - is used to run any query.

B. CallableStatement - is used to call stored procedures in a database.

C. FutureStatement - is a class that returns results in the future.

D. AsyncStatement - is a class to work with databases in asynchronous mode.

E. PreparedStatement - is used to run queries in a more safe way.

1. A
2. B
3. C
4. D
5. E
6. None of them

Correct Answer: 1, 2, 5

Explanation:

In Java jdbc , you will work with a Statement, PreparedStatement, or CallableStatement. PreparedStatement and CallableStatement are extended from Statement. FutureStatement and AsyncStatement does not exists.

- ➢ https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

36. What is the output of the code below?

```
public class Tree {
   int age;
   public void getAge() {
      System.out.print(" Tree age = " + age);
   }
}
....
public class Oak extends Tree {
   int age;
   public Oak(int age) { this.age = age; }
   public void getAge() { System.out.print(" Oak age
= " + this.age);}
}


....
   public static void main(String[] args) {
      Tree s = new Oak(40);
      System.out.print(s.age);
      s.getAge();
   }
```

1. 40 Oak age = 40
2. 40 Tree age = 40
3. 0 Oak age = 40
4. 0 Tree age = 0

5. 40 Tree age = 0
6. 40 Oak age = 0

Correct Answer: 3

Explanation:

In java when we create variable with reference type of Parent and Child object type, we access to parent instance fields but child methods. So in our case field 'age' of Tree equals 0 but Oak method returns 40.

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

37. What is the output of the code below?

```
public class Parent {
    public int weight;
    public void increment() { weight = weight + 1;}
    public void getWeight() {
System.out.print(weight); }
}
public class Child extends Parent{
   public Double weight;
   public Child(Double weight, Integer
weightOfParent) {
        this.weight = weight;
        super.weight = weightOfParent;
    }
    public void getWeight()
{System.out.println(weight); }
}
```

```
public class Main {
   public static void main(String[] args) {
      Parent p = new Child(30.5, 75);
      p.increment();
      System.out.print(p.weight);
      p.getWeight();
   }
}
```

1. 7676
2. 31.575
3. 7575
4. 30.576
5. 7630.5
6. None of them

Correct Answer: 5

Explanation:

Option 5 correct. By definition instance variables are chosen from the reference type, not instance type, and polymorphism is not applicable to variables.

When we call p.increment() it will increase parent instance field because child class do not have such method. Just because variables are not overriden when we call System.out.print(p.weight); it will show us value of parent field. When we call p.getWeight() it will print 30.5 because child class overrides method getWeight();

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

38. What is the output of the code below?

```
public class Predator {
   private int age;
   public Predator() {  System.out.print("Predator"); }
}

public class Lion extends Predator{
   public Lion() { System.out.print("Lion"); }
}

public class Cub extends Lion{
   public Cub() { System.out.print("Cub"); }
}

public class Main {
   public static void main(String[] args) {
      new Lion();  System.out.println();
      new Cub();
   }
}
```

1. PredatorLionCub
   PredatorLionCub

2. Predator
   PredatorLion

3. PredatorLionCub
   PredatorLion

4. PredatorLion
   PredatorLionCub

5. none of them

Correct Answer: 4

Explanation:

Java will create instance of class and also will load its parents. In our case when we call new Lion() its parent Predator is also created, when we call new Cub() then Lion, Predator classes are loaded.

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

39. What is the output of the code below?

```
public class Country {
   private int cityCount;
   public Country(int cityCount) {
      this.cityCount = cityCount;
      System.out.println("Country"); }
   public int getCityCount() {
      return cityCount;}
}

public class EuropeCountry extends Country {
   private int count;
   public EuropeCountry(int count) {
      this.count = count;
      System.out.println("EuropeCountry");
   }
   public int getCityCount() { return 0; }
}

public class Main {
   public static void main(String[] args) {
        new Country(23);
    }
}
```

1. EuropeCountry
2. EuropeCountryCountry
3. CountryEuropeCountry
4. code will not compile
5. none of them

Correct Answer: 4

Explanation:

Code will not compile, because there is no default constructor available in Country. Or we need to add in EuropeCountry constructor code super(count);

- ➢ https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

40. What is the output of the code below?

```
public class Tree {
    private void setAge(int age) {
        System.out.print("private " + age);
    }
    protected void setAge(double age) {
        System.out.print("protected " + age);
    }
    void setAge(long age) {
        System.out.print("default " + age);
    }

    public static void main(String[] args) {
        Tree t = new SubTree();
        t.setAge(1);
        t.setAge(1.0);
        t.setAge(1l);
    }
}
class SubTree extends Tree {
    public void setAge(int age) {
        System.out.print("child public " + age);
    }
    protected void setAge(long age) {
        System.out.print("child protected " + age);
    }
        void setAge(double age) {
            System.out.print("child default " + age);
        }
}
```

1. private 1protected 1.0child protected 1
2. child private 1child protected 1.0child protected 1
3. private 1protected 1.0protected 1
4. child private 1protected 1.0protected 1
5. code will not compile

Correct Answer: 5

Explanation:

Code will not compile because we are attempting to assign weaker access privileges ('package-private') was 'protected' to method setAge(double age) in SubTree class;

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

Control Flow (Loop, switch, if/else)

41. What is the output of the code below?

```
int age = 10;
var weight = (age++) + 2;
if(weight > 13) { System.out.print("weight is 13");}
else
if((++age)*weight%2 ==
0)System.out.println("even");
else System.out.println("odd");
```

1. odd
2. even
3. weight is 13
4. weight is 13even
5. weight is 13odd
6. code will not compile

Correct Answer: 2

Explanation:

Option 2 is correct.

Code will compile.

var weight = 12;

so it goes to else statement

which has its own if else.

++age = 11, 11*12 = 132 which is even number.

so output will be 'even'

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

Date/Time API

42. What is the output of the code below?

```
var now = LocalTime.now();
now = now.plus(Period.of(1,0,0));
now = now.plusHours(2);
System.out.println(now);
```

1. current time in default timezone + 2 hours and 1 year
2. current time in UTC-00 + 2 hours and 1 year
3. code will not compile
4. will throw UnsupportedTemporalTypeException
5. none of them

Correct Answer: 4

Explanation:

Option 4 correct. will throw UnsupportedTemporalTypeException, because we can't add Period to LocalTime.

- ➢ https://docs.oracle.com/javase/tutorial/datetime/iso/overview.html

43. What is the output of the code below?

```
var startDate = LocalDate.of(2023, 2 , 1);
var endDate = LocalDate.of(2023, 0 , 1);
```

```
System.out.println("between : " +
Period.between(startDate, endDate));
```

1. between : P-1M
2. between : P-2M
3. code will not compile
4. will throw RuntimeException

Correct Answer: 4

Explanation:

Option 4 correct. Month starts from 1 not from 0. That’s why code will throw exception.

- ➢ https://docs.oracle.com/javase/tutorial/datetime/iso/overview.html

Arrays and Collections

44. What is the output of the code below?

```
var list = new ArrayList<String>();
list.add("A");
list.add("B");
var copy = list;
copy.add("C");
System.out.print(copy.equals(List.of("A", "B", "C")));
var ulist = List.of("A", "B");
ulist.add("C");
System.out.print(list.equals(ulist));
```

1. truetrue
2. truefalse
3. falsefalse
4. falsetrue
5. code will throw exception

Correct Answer: 5

Explanation:

List.of(varargs) - creates unmodifiable collection, any modifications will throw exception.

- ➢ https://docs.oracle.com/javase/tutorial/collections/intro/index.html

45. Select correct statement that will output items of map sorted by their values.

```
var map = Map.of(1, "A", 2, "B");
```

1.
   ```
   Comparator<Map.Entry<Integer, String>> cmp = Comparator.comparing(Map.Entry::getValue);

   map.entrySet().stream().sorted(cmp).forEach(i -> {
              System.out.println("sk=" + i.getKey() + " - v=" + i.getValue());
          });
   ```

2.
   ```
   map.entrySet().stream().forEach(i -> {
              System.out.println("k=" + i.getKey() + " - v=" + i.getValue());
          });
   ```

3.
   ```
   map.entrySet().stream().sorted().forEach(i -> {
              System.out.println("sk=" + i.getKey() + " - v=" + i.getValue());
          });
   ```

4.
   ```
          BiConsumer<Integer, String> biConsumer = (k, s) -> {
              System.out.println("k=" + k + " - s=" + s);
          };
          map.forEach(biConsumer);
   ```

5. none of them

Correct Answer: 1

Explanation:

Map is not ordered. We need to call sorted method with Comparator as a parameter. Option 2, 3 will simply output map without order. Option 4 will throw exception 'class java.util.KeyValueHolder cannot be cast to class java.lang.Comparable'. Option 5 incorrect.

- https://docs.oracle.com/javase/tutorial/collections/intro/index.html

46. Select statements that are true about collections in java.

A. TreeSet elements are stored in a sorted tree structure.

B. Set can contain duplicates.

C. LinkedList items can be accessed by their indices.

D. HashMap can contain null as a key.

E. HashSet elements are sorted in a natural order

1. A
2. B
3. C
4. D
5. E

Correct Answer: 1,3,4

Explanation:

A correct. TreeSet elements are stored in a sorted tree structure.

B incorrect. Set do not contain duplicates.

C correct. LinkedList items can be accessed directly using index.

D correct.. HashMap can contain null as a key.

E incorrect. HashSet elements are not sorted.

- https://docs.oracle.com/javase/tutorial/collections/intro/index.html

Concurrency

47. What is the output of the code below?

```
public class ThreadMain {
   public static void main(String[] args) {
      AtomicInteger atomicInteger = new
AtomicInteger(0);
      var t1 = new Thread(() -> {
         atomicInteger.incrementAndGet();
      });
      t1.run();
      System.out.print("t1 :" + atomicInteger.get());
      var t2 = new Thread(() -> {
         atomicInteger.incrementAndGet();
      });
      t2.run();
      System.out.print(", t2 :" + atomicInteger.get());
   }
}
```

1. t1 :0, t2 :0
2. t1 :1, t2 :2
3. t1 :1, t2 :1
4. t1 :0, t2 :1
5. none of them

Correct Answer: 2

Explanation:

Option 2 is correct. When we call the run() method of the Thread class it will be executed in the current thread not asynchronously. So Threads are executed in the main thread sequentially and values are 1 and 2.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

48. What is the output of the code below?

```
public class DaemonThread {
   public static void main(String[] args) {
      Runnable lambda = () -> {
          try {Thread.sleep(2000);
            System.out.print("-lambda-");}
         catch (InterruptedException e) { throw new
RuntimeException(e);}
      };
      var d = new Thread(lambda);
      d.setDaemon(true);
      d.start();
      var t = new Thread(() -> {
         System.out.println("-t-");
      });
      t.start();
   }
}
```

1. -t-

2. -t--lambda-

3. -lambda-

4. -lambda--t-
5. code will not compile

Correct Answer: 1

Explanation:

Option 1 correct. Thread d is a daemon thread and java will finish execution before it outputs to console.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

49. What is the output of the code below?

```
public class ParallelStreamOut {
  public static void main(String[] args) {
    List<Integer> list = List.of(1,6,2,10,7,8);
    list.stream()
        .parallel()
        .limit(4)
        .forEach(System.out::print);
  }
}
```

1. 1,6,2,10
2. 6,2,10,7
3. 2,10,7,8
4. 2,10,7,8
5. none of them

Correct Answer: 5

Explanation:

Since the stream executed in parallel order of items is random. We can™t predict how items will be outputted to the console.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
- https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html

50. What is the output of the code below?

```
public class CallableSample implements
Callable<String> {
   public static String amount;
   @Override
   public String call() throws InterruptedException {
      Thread.sleep(1000);
      return amount = amount + 1;
   }
}

public class Main {

   public static void main(String[] args) throws
Exception {
      ExecutorService service=
Executors.newFixedThreadPool(2);
      CallableSample cs = new CallableSample();
      Future<String> future = service.submit(cs);
      service.execute(() -> {
         try {
            String result = future.get(100,
TimeUnit.MILLISECONDS) + "2";
            System.out.println(result);
         } catch (Exception e) {
System.out.println("exception"); }
      });
   }
}
```

1. 12
2. null1exception
3. exception
4. null12
5. none of them

Correct Answer: 3

Explanation:

Submitted CallableSample will sleep for 1 second, second task will wait for the result of the first task for 100 milliseconds and then throws an exception.

- https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
- https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/ExecutorService.html

Concurrency

1. What is the output of the code below?

```
public class Main extends Thread {
   public static int count = 0;
   public static void main(String[] args) {
      Main thread = new Main();
      thread.start();
      while(true) {
         if(thread.isAlive() == false) {
            count++;
            System.out.println("Main: " + count);
            break;
         }
      }
   }
   public void run() {
      count++;
   }
}
```

1. 1
2. 0
3. 2
4. will loop forever
5. will not compile

Correct Answer: 3

Explanation:

Code is correct. Thread will run and increments count. isAlive() will check if the current thread is still alive. When it ends execution we increment count one more time and break the while(true) loop then the program stops execution.

2. What is the output of the code below?

```
public class AtomicMain {
   public static final AtomicInteger atomicInteger =
new AtomicInteger(0);
   public static void main(String[] args) {
      var t1 = new Thread(() -> {
         atomicInteger.getAndIncrement();
         System.out.print(" t1 :" + atomicInteger.get());
      });
      t1.start();
      var t2 = new Thread(() -> {
         atomicInteger.getAndIncrement();
         System.out.print(" t2 :" + atomicInteger.get());
      });
      t2.start();
   }
}
```

1. t2 :2 t1 :1
   or t2 :1 t1 :2

2. t2 :1 t1 :2

3. t2 :2 t1 :4

4. code will not compile
5. t2 :2 t1 :1
6. none of them

Correct Answer: 1

Explanation:

t1 and t2 are executed in different threads and we can't predict which of them will be executed first. AtomicInteger will be incremented before it goes to output. So values will be 2,1 or 1,2.

3. Fill in the blanks.

__ is a situation when a thread doesn't execute completely in a quantum of time and its state is saved by thread scheduler.

__ is a special type of variable that can increment its value by using low-level machine instructions such as compare-and-swap.

1. Context Switch, Volatile
2. Context Switch, AtomicInteger
3. Livelock, Volatile
4. Race condition, AtomicInteger
5. Race condition, Volatile

Correct Answer: 2

Explanation:

Context switching - is the technique where CPU time is shared across all running processes and processor allocation is switched in a time bound fashion. To schedule a process to allocate the CPU, a running process is interrupted to a halt and its state is saved.

Atomic variables - use low-level algorithms to increment its value atomically.

Volatile is java variable type that is opposite to final, meaning that its value can be changed. It does not increment its value.

Livelock is the case when 2 threads are active and waiting for response of other thread.

Race condition is the case when multiple threads execute block of code concurrently but suppose to use sequentially.

Var

4. Select statements that are true about var.

A. can be used as an instance field type.

B. can be a parameter type of method.

C. can™t be initialized with null.

D. can™t change its type to another after initialization.

E. we can create the variable name var of type var.

1. A
2. B
3. C
4. D
5. E

Correct Answer: 3,4,5

Explanation:

1 invalid. Var - can't be used as an instance field type because it was designed to use in methods or constructors.

2 invalid. Var – can’t be a parameter type of method because it needs to be initialized.

3 valid. Var – can’t be initialized with null.

4 valid. When we create a variable of type var and initialize it with integer then we can™t change it to the string. Var type is not dynamic like in js.

5. valid. Currently java allows to create variable var of type var.

5. Select correct statements.

```
3  public class VarSample {
4    Integer count;
5    public void init(var a) {
6        String name;
7        var b = count;
8        var s = name;
9        var k = 1;
10       k = "one";
11   }
12 }
```

1. code is valid
2. line 5 cause compile error
3. line 7 cause compile error
4. line 8 cause compile error
5. line 10 cause compile error
6. none of them

Correct Answer: 2, 4, 5

Explanation:

Code is invalid.

Option 2 is valid. Line 5 will cause a compile error, because var can™t be a type of method parameter.

Option 3 is invalid. Line 7 will work. Instance variables are initialized by default. Variable b will be of type Integer with null value.

Option 4 is valid. Line 8 will cause a compile error, because String name is not initialized and we are trying to assign it to var s.

Option 5 is valid. Line 10 will cause a compile error, because var k has got its type and trying to change from integer to string will cause a compile error.

String

6. What is the output of the code below?

```
var java = "java";
var oneMore = new String("java");
boolean b1 = "java" == java;
boolean b2 = java == oneMore;
boolean b3 = "java " == java + " ";
System.out.print(b1);
System.out.print(b2);
System.out.print(b3);
```

1. truefalsetrue
2. truefalsefalse
3. falsetruefalse
4. truetruetrue
5. truetruefalse

Correct Answer: 2

Explanation:

b1 is true because var java is referencing value from string pool and when we compare "java" - which is taken from string pool and variable java it will be true.

b2 is false because var java is literal but oneMore is Object thats why they have different reference value.

b3 is false because concatenation of java + " " is done in runtime and its considered as a new string thats why it has no reference to value from string pool. so comparison of different references will result false

7. What is the output of the code below?

```
public class Main {
    public static void main(String[] args) {
        var a = "aa";
        String b = a + "b", c="aabb";
        Main m = new Main();
        if(b == (a  + "b")) {
            m.runSwitch(a);
            m.runSwitch(b);
        } else {
            m.runSwitch(c);
            if(c == new String("aabb").intern()) {
                System.out.println("c==d");
            }
        }
    }
    public void runSwitch(String a) {
        var aa = switch (a) {
            case "aa" -> "equals aa";
            case "aab" -> "equals aab";
            default -> "default";
        };
        System.out.print(aa + " ");
    }
}
```

1. code will not compile
2. equals aa equals aab

3. equals aa
4. default
5. default c==d

Correct Answer: 5

Explanation:

b == (a + "b") will be false. Since b and (a + b) have + concatenation operations they will be created as new strings at runtime and will not reference the same string in the string pool.

java will go toelse™ statement.

In the else statement condition c == new String("aabb").intern() will return true.

After calling intern() on a String object java will look if the string pool contains the same string and return reference to it otherwise will insert a new string in a pool. Since string pool already contains aabb comparison of 2 equal references will be true.

8. Select correct statement.

A. String is immutable.

B. StringBuffer and StringBuilder are mutable.

C. StringBuilder is a synchronized version of StringBuffer.

1. A,C
2. B, C
3. A, B, C
4. A, B

5. A

Correct Answer: 4

Explanation:

String is immutable so when we call methods like replace or toUpperCase the new string is created.

StringBuffer and StringBuilder are mutable, you can add, replace or delete items but it will not create new objects of StringBuffer or StringBuilder.

Streams

9. What is the output of the code below?

```
public class Q5main {
    public static void main(String[] args) {
        var items = List.of(1, 4, 6, 3, 9);
        Stream.iterate(0, i -> i + 1).limit(5).forEach(i ->
items.add((int)(Math.random()*10)));
        System.out.println(items.size());
        items.stream().filter(i -> i%2==0).forEach(i -> {
            System.out.print(i);
        });
    }
}
```

1. 10
   46 and even elements from last 5 items
2. runtime exception
3. 9
   46 and even elements from last 5 items
4. 10
5. none of them

Correct Answer: 2

Explanation:

Code will throw runtime exception because we are trying to modify unmodifiable List.of(1,4,6,3,9). We can't add or delete it's contents.

- https://docs.oracle.com/javase/tutorial/collections/streams/index.html#pipelines
- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

10. What is the output of the code below?

```
public class Q52Main {

   public static void main(String[] args) {
      var list1 = List.of("one", "two", "three");
      var list2 = List.of("1", "3", "4", "11");
      var main = List.of(list1, list2);
      var max = main.stream().flatMap(i ->
i.stream()).filter(i -> i.length() == 1).mapToInt(i ->
Integer.parseInt(i)).max();
      System.out.print(max.getAsInt());

      var ml = main.stream().flatMap(i ->
i.stream()).map(i -> i.length()).mapToInt(i ->
i).max();
      System.out.print("-" + ml.getAsInt());
   }
}
```

1. 4-Mar
2. 3-May
3. 4-May
4. 11-May
5. none of them

Correct Answer: 3

Explanation:

The first stream will filter strings whose length equals 1, then it converts from string to integer and gets max value = 4.

The second stream will map a stream of maps to a stream of integers by simply taking their length then we get a string with max length which is 5.

- https://docs.oracle.com/javase/tutorial/collections/streams/index.html#pipelines
- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

11. What is the output of the code below?

```
public class Q3 {
   public static void main(String[] args) {
      var stream = Stream.iterate(1, (s) -> s + 5);
      var max= stream.limit(5).mapToInt(i ->
i).max();
      var any= stream.filter(i -> i % 3 == 0).findFirst();
      System.out.print("max:"+max);
      System.out.print(" any:"+any);
   }
}
```

1. compile error
2. runtime error

3. max:OptionalInt[21] any:Optional[6]
4. max:21 any:6
5. max:20 any:3

Correct Answer: 2

Explanation:

java.lang.IllegalStateException: stream has already been operated upon or closed. Because we are trying to use same stream twice.

- https://docs.oracle.com/javase/tutorial/collections/streams/index.html#pipelines
- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

12. What is the output of the code below?

```
public class Q4 {

   public static void main(String[] args) {

      Predicate<String> pred = (i) -> !i.isEmpty();
      StringBuilder str = new StringBuilder();
      var streamOne = Stream.generate(() -> "2");
      var streamTwo =
Stream.generate(Math::random);
      var anyMatch = streamOne.anyMatch(pred);
      str.append(anyMatch);
```

```
        var noneMatch =
streamTwo.limit(2).noneMatch(Predicate.isEqual(2.
0));
        str.append(noneMatch);
        System.out.println(str);
    }

}
```

1. truetrue
2. falsetrue
3. truefalse
4. code will hang
5. none of them

Correct Answer: 2

Explanation:

anyMatch - Returns whether any elements of this stream match the provided predicate. May not evaluate the predicate on all elements if not necessary for determining the result. If the stream is empty then false is returned and the predicate is not evaluated. Since our stream returns 2 every time then the result will be true.

noneMatch - will take the first 2 elements from the stream and then noneMatch() will check if there is an element which may match. Since stream values are always between 1 and 0 none of the items will match so the result will be true.

- https://docs.oracle.com/javase/tutorial/collections/streams/index.html#pipelines

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

13. Select method(s) that will make the stream output its items to the console.

1. var stream = Stream.of(12,4,5,6,2);
   stream.mapToInt(i -> i).peek(System.out::println);

2. multiple-choice

3. distinct()

4. findAny()

5. forEach(i->i)

6. count()

7. reduce(-1 , (a,b) -> a + b)

Correct Answer: 5

Explanation:

- Distinct() is intermediate operation but stream needs terminal operation to be executed.

- findAny() -gets any element from stream and usually its a first element thats why whole stream will not be traversed. It will output only one element to console.

- forEach(i->i) - is a compile error, must be forEach(i -> {})

- count() - is terminal operation which can get size of stream without executing intermediate operations. So peek will not output items.
  reduce - will traverse the stream and the peek will output all items to the console.

- https://docs.oracle.com/javase/tutorial/collections/streams/index.html#pipelines

- https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html

Records

14. Select statements that are true about record.

1. record class can't implement sealed interface
2. record class is simple data carrier
3. has public access methods and private final fields
4. record class is immutable
5. record can have native method

Correct Answer: 2,3,4

Explanation:

1 - invalid. Record can implement the sealed interface.

2 - valid. Record is created to safely carry data.

3 - valid. Its instance fields must by final and private to be immutable. But has public access methods.

4 - valid. Record is immutable, it has final and private fields with no setter methods.

5-invalid. Record can extend sealed class as a normal class in java.

- ➢ https://docs.oracle.com/en/java/javase/17/language/records.html

14. Select statements that is true about record.

1. we can create new non-static instance fields
2. record class can't extend sealed class
3. record can not have inner class as usual java class
4. has private methods and private final fields
5. new instance of record class can be created using new keyword

Correct Answer: 5

Explanation:

1-invalid. Record can't have the non-static instance fields.

2-invalid. It can extend the sealed class.

3-invalid. It can have the nested classes.

4-invalid. Public access methods and private final fields.

5-true. We can create new instance of record by using keyword new.

- https://docs.oracle.com/en/java/javase/17/language/records.html

Sealed Classes

16. Select statement that is true about sealed classes.

1. when child classes are located in the same file as the parent, we can omit the keyword 'permits'.
2. children may or may not extend the parent
3. children of the sealed classes can't be extended
4. children of sealed class can't implement the interface
5. none of the above

Correct Answer: 1

Explanation:

1 valid. If Parent and child class located in same file, we can omit keyword 'permits' in Parent class declaration.

2. invalid. Children must extend sealed Parent.

3. invalid. Children of sealed class can have its own children classes.

4 invalid. Children of sealed class can impement interface

- ➢ https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- ➢ https://openjdk.org/jeps/411

17.Select statements that are true about code below.

```
public final sealed class SParent permits SChild {
   public void hello() {
      System.out.println("Hello from Parent");
   }
   public SParent(String name) {
   }
}

public sealed class SChild extends SParent{
   public SChild(String name) {
      super(name);
   }
}
```

1. code will compile
2. remove keyword final in parent class declaration to make this class compile.
3. Child class will compile if we change sealed modifier to non-sealed in child class
4. Child class will compile if we remove keyword sealed from declaration
5. none of them

Correct Answer: 2,3

Explanation:

1 invalid. Code has errors.

2 valid. Final class can't be extended.

3 valid. Child class does not permits to other classes to extend itself that's why we need to change it to non-sealed.

4 invalid. Children of sealed class must be of type final, sealed or non-sealed.

5 invalid.

- https://docs.oracle.com/en/java/javase/17/language/sealed-classes-and-interfaces.html;
- https://openjdk.org/jeps/411

Arithmetic/Boolean Expressions (Operators, Promotion, Casting)

18. What is the output of the code below?

```
public class Main {
    public static int a = 0;
    public static void main(String[] args) {
        var b = 7;
        int c = b+++a;
        var d = --c + (++a) - b;
        System.out.print(c + "-" + b + "-" + d);
    }
}
```

1. 6-8--1
2. code will not compile
3. 7/7/2002
4. 7/7/2000
5. 8/7/2001

Correct Answer: 1

Explanation:

int c = b+++c can be rewritten into (b++) + c = 7 since b is using post increment it will return its value and then get incremented.

so the value of b in the next operation will be 8.

var d= --c +(++a) -b;

--c - this is the example of pre-decrement so it will decrement and then return value, so result of c will be 6.

(++a) = will be 1 because it will increment and then return the value which becomes 1.

var d = 6 + 1 - 8 = -1

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

19. What is the output of the code below?

```
public class Q2 {
   public static void main(String[] args) {
      var d = 11; var k =-1;
      var c +=d;
      double dd = d * 9.0 + c/-1 + (--k);
      var res = (int)dd % 4;
      System.out.println(res);
   }
}
```

1. code will not compile
2. 3
3. 0
4. 1
5. 2

Correct Answer: 1

Explanation:

Code will not compile because of   var c +=d;

c is not initialized thats why why cant use it.

Consider it as var c = null + d.

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

20. Select statement(s) that are true about code below.

```
public class Q3 {
   public static void main(String[] args) {
      int a = 11; short b = 9;
      long c = 8; double f = 3;
      var r1 = a*b;
      var r2 = c+r1;
      var r3 = r2 + f;
   }
}
```

1. r1 is int
2. r2 is int
3. r2 is long
4. r3 is long
5. r3 is double
6. r1 is short

Correct Answer: 1, 3, 5

Explanation:

in java result of operation of different types will be of larger type.

(int) operation (short) = (int) because int has more size than short

(int) operation (long) = (long) because long has more size than int

(long) operation (double) = (double) because long has more size than long

r1 - int, r2 - long, r3 -double

- https://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html

21. What is the output of the code below?

```
public class Q4 {
   public static void main(String[] args) {
      var cond = false;
      int a = 7;
      var b = (byte)(Byte.MAX_VALUE + 1);
      int k = 2;
      k--;
      a = a + b;
      System.out.println(~k + " | " + a + " | " +
(int)cond);
   }
}
```

1. -2 | -121 | false
2. -2 | 137 | 0
3. -2 | -121 | 0
4. -1 | 137 | 0
5. code will not compile

Correct Answer:

Explanation:

Code will not compile because we cant cast boolean to integer. As we can't cast var b = (boolean) 0;

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html

Math API

22. What is the output of the code below?

```
public class MathApi {
   public static void main(String[] args) {

       var random = Math.random();
       var ternary = random == 0.0 ? 0.1 : random;
       var ceil = Math.ceil(ternary);
       var pow = Math.pow(3, 2) * ++ceil;
       System.out.print(pow);
       System.out.print("-");
       var sqrt = Math.sqrt(25);
       var result = sqrt * pow % 2;
       System.out.print(result);
   }
}
```

1. 18.0-1.0
2. 9.0-0.0
3. 18.0-0.0
4. 9.0-1.0
5. none of them

Correct Answer: 3

Explanation:

Math.random() returns double in range 1 > i >=0 and if the value of random is 0 then we return 0.1

Math.ceil will convert any number to 1 that comes from the variable ternary.

var ceil = 1.0 always.

var pow = Math.pow(3, 2) * ++ceil;

Math.pow(3,2) is the 3 to the power of 2 = 9.

++ceil = will increment and return value so at this moment ceil will return 2.

var pow = 9.0 * 2.0 = 18.0

var sqrt = Math.sqrt(25); takes the square root of the number which is 5.

var result = 5.0 * 18.0 % 2 = 90.0%2 = 0.0

- https://docs.oracle.com/javase/tutorial/java/nutsandbolts/operators.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html

Lambda

23. Select valid option(s) of the functional interface.

1.
   ```
   @FunctionalInterface
   public interface F1 {
      void sayHi();
   }
   ```

2.
   ```
   public interface F2 {
      void getName();
   }
   ```

3.
   ```
   @FunctionalInterface
   public interface F3 {
      void getName();
      void sayHi();
   }
   ```

4.
   ```
   public interface F4 {
      void getName();
      static void sayHi() {}
   }
   ```

5.
   ```
   public interface F5 {
      void getName();
      default void sayHi() {}
   }
   ```

Correct Answer: 1,2,4,5

Explanation:

Functional interface - has a single abstract method. Annotation @FunctionalInterface is optional. It can have default and static methods; they will not be considered as a second abstract method.

1 valid. it has appropriate syntax.

2 valid. also appropriate.

3 invalid. it contains 2 abstract methods, but only one allowed.

4,5 are valid and appropriate.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/function/package-summary.html

24. Select correct usage of the FunctionalInterfaces.

Predicate<Integer> p = i -> i%2 == 0; // A

BiFunction<Integer, Double, Integer> bi = (a, b) -> { var c = a + b; return c;}; // B

BiConsumer<Integer, Double> bic = (a, b) -> { var c = a+b; }; // C

Supplier<LocalDateTime> s = () -> { LocalDateTime.now(); }; // D

1. A

2. B

3. C

4. D

5. none of them

Correct Answer: 1,3

Explanation:

1 valid.Predicate - used for boolean statements.

2 invalid. BiFunction consume 2 parameters and return result. Problem is that it consume Int and Double and return Integer rather than Double. Sum of Double and Integer will be Double.

3 valid. BiConsumer - consumes and returns nothing.

4 invalid. Supplier nothing consumes but returns value. In this sample we need return LocalDateTime.

5. Invalid.

- https://docs.oracle.com/javase/8/docs/api/java/util/function/Supplier.html
- https://docs.oracle.com/javase/8/docs/api/java/util/function/BiConsumer.html
- https://docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html
- https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html

25. What is the output of code below?

```
package lambda.q5.three;
import java.util.function.Consumer;
public class Main {
   public static void main(String[] args) { new
Main().method(); }
   public void method() {  consume((var var) -> {});
}
   public void consume(Consumer<String> var) {
      System.out.println(var);
   }
}
```

1. null
2. lambda.q5.three.Main$$Lambda
3. will not compile because of var
4. none of them

Correct Answer: 2

Explanation:

Code will compile and run. Since we passed the lambda expression, then output is lambda.

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/function/package-summary.html

Localization

26. Select order of steps that java will do to get resources from ResourceBundle?

```
ResourceBundle bundle =
ResourceBundle.getBundle("rbundles", new
Locale("en",  "US"));
var name = bundle.getString("name");

key 'name' does not exists in any file.

A. Look for the resource with the same language
code.
B. Look for the resource with the same language
code and country code.
C. Look for the default resource bundle.
D. Look for the resource with the same language
code but different country code.
E. Throws Runtime Exception.
```

1. B, C, E
2. B, D, A, C, E
3. B, D, A, C
4. B,A,C,E
5. none of them

Correct Answer: 4

Explanation:

Firstly it will go to exact match property which rbundles_en_US.properties, if not finds then goes to rbundles_en.properties, if not finds then goes to rbundles.properties if not finds then throws runtime exception.

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ResourceBundle.html

27. Select statements that are true about Localization in java.

1. enCA is valid Locale identifier
2. language code and country must be separated by underscore
3. ResourceBundle gets its translations from table.
4. ResourceBundle works with .properties files
5. When key is not exist in properties, java will fail silently without throwing an exception.

Correct Answer: 2, 4

Explanation:

1 invalid. en_CA is valid Locale Identifier.

2 valid.

3 invalid. ResourceBundle gets the value by the key from .properties (Ex: hello_en_CA.properties)

4 is valid.

5. When key requested is not exists in .properties then Runtime exception is thrown.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ResourceBundle.html

Modules

28. Program has 3 modules named m.one, m.two and m.three. Module m.two depends on module m.three and it alreadyrequires™ m.one module. What we must add to module-info.java in module m.one to resolve the situation.

1. requires m.three;
2. requires transitive m.three;
3. requires static m.three;
4. open m.three;
5. none of them

Correct Answer: 2

Explanation:

requires transitive - used to require module and also make it available for other modules that could potentialy 'requires' current module.

requires - do not have such functionality, it only makes exported packages available for current module.

requires static - used to require modules at compile time and optional for runtime.

open - is used to open module for reflection mechanism.

- https://www.oracle.com/corporate/features/understanding-java-9-modules.html

29. Select statements that are true about modules in java.

A. Keyword exports <module_name> - used to export module, make it available for other modules;

B. Keyword open - used to open module for reflection mechanism;

C. Keyword requires <module_name> - used when we need some class or method in another module.

D. Keyword exports <package_name> - used to export package;

1. A
2. B
3. C
4. D

Correct Answer: 2,3,4

Explanation:

requires - do not have such functionality, it only makes exported packages available for current module.

open - is used to open module for reflection mechanism.

exports is used to export package functionality. Keyword exports can't be used with module name.

- https://www.oracle.com/corporate/features/understanding-java-9-modules.html

30. By using this tool developer can create custom build of java.

1. jmods
2. jdeps
3. jlink
4. jcreator
5. jre_mods

Correct Answer: 3

Explanation:

Jlink is special tool for creating custom java builds. We can include only needed modules and decrease size of it.

31. Select statements that are valid.

```
module m.one {
  exports my.dummy.package;
  requires m.two;
  requires transitive m.two;
}
```

1. java.desktop - contains gui packages
2. The named module can contain children and the parent sealed class in different packages.
3. keyword exports used alongside module name.

4. automatic module does not contain module-info.java

Correct Answer: 2,3,5

Explanation:

1 invalid. Could not 'requires' 2 times of same module.

2 valid. java.desktop - contain GUI packages.

3 valid. Parent sealed class and its children can locate in different packages in named module.

4 invalid. exports <package-name> not <module-name>

5 valid. An automatic module appears on the module path but does not contain a module-info.java file. It is simply a regular JAR.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Module.html
- https://jenkov.com/tutorials/java/modules.html#automatic-modules

JDBC

32. Select the correct option.

1. Used to run stored procedure

2. More preferable to run queries safely.

3. using it can end with sql injection

4. Allows the use of a special method to set parameters of query.

1. 1. StoredStatement
   2. Statement
   3. PreparedStatement
   4. CallableStatement

2. 1. CallableStatement
   2. PreparedStatement
   3. Statement
   4. PreparedStatement

3. 1. CallableStatement
   2. AsyncStatement
   3. Statement
   4. FutureStatement

4. 1. CallableStatement
   2. PreparedStatement
   3. CallableStatement
   4. PreparedStatement

5. none of them

Correct Answer: 2

Explanation:

CallableStatement - used to call stored procedure in database.

PreparedStatement - preferable to run sql it's more secure and optimized. We can set query parameters using special method that saves from sql injection.

Statement - used to run plain sql query string. Potentially open for sql injection.

1 invalid. No such Class like StoredStatement.

2 valid.

3 invalid. No such class like AsyncStatement.

4 invalid. Sql injection is more likely to happen with Statement, because it does not have special parameter setting methods.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/Statement.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/CallableStatement.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/java/sql/PreparedStatement.html

33. Select option that helps to run 2 update queries in transaction using JDBC.

1. 1. conn.setAutoCommit(false)
   2. conn.setAutoCommit(true)

2. 1. conn.setAutoCommit(false)
   2. conn.commit()

3. 1. conn.setAutoCommit(false)
   2. conn.commit(true)

4. 1. conn.commit(false)
   2. conn.commit(true)

5. conn.setAutoCommit(false)

Correct Answer: 2

Explanation:

setAutoCommit(false) - will disable the commit of the query after being executed. No sql statements are committed until you call method commit() explicitly.

Ex:

```
setAutoCommit(false)
run query1
run query2
commit()
```

1 invalid. Does not commit changes.

2 valid.

3 invalid No such method commit() with boolean parameter.

4 invalid. Did not call setAutoCommit(false).

5 invalid. Need commit() method.

- https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html

Java OOA GC

34. What is the output of code below if System.gc() is executed successfully?

```
public class GCSample {
    public String name;
    public GCSample(String name) {
        this.name = name;
    }
    public static void main(String[] args) throws
InterruptedException {
        var a = new GCSample("a");
        var b = new GCSample("b");
        var c = b;
        b = null;
        new GCSample("empty");
        a = null;
        System.gc();
        Thread.sleep(2000);
    }
    @Override
    protected void finalize() throws Throwable {
        System.out.println("destroy : " +this.name);
    }
}
```

1. code will not compile
2. destroy : a

3. destroy : empty
   destroy : a

4. destroy : b
   destroy : a

5. destroy : b
   destroy : empty

Correct Answer: 3

Explanation:

GCSample("empty") - created without reference thats why when an object has lost all its references it is eligible for gc.

GCSample("a") has reference 'a' but when code executed a = null this objects lost its reference and become eligible for gc.

GCSample("b") has reference 'b', then it also got reference named 'c'. When b = null executed this object is still had reference 'c' that's why it is not eligible for gc.

1 invalid code will compile.

2.invalid. Must be also "destroy : empty"

3 valid.

4 invalid GCSample("b") is not eligible.

5 invalid. GCSample("b") is not eligible.

- https://www.oracle.com/technetwork/tutorials/tutorials-1876574.html

35. Select statements that are true about garbage collection(gc).

1. System.gc() will immediately clean eligible objects.
2. Object that has at least one reference is not eligible for gc.
3. Primitives and Objects are eligible for gc when they lose all their references.
4. When Object and their references are out of scope they become eligible for gc.
5. Immutable class is not eligible for gc.

Correct Answer: 2,4

Explanation:

1 invalid. System.gc() does not guarantee the execution of cleaning the eligible object.

2 valid.Object is eligible for gc when does not have references.

3 invalid. Primitives are not cleaned by gc. Because they are not objects.

4 valid. If object that is created in some method is executed and its reference become out of scope, they become eligible for gc.

5 invalid. Immutable class can also become eligible for gc.

- ➢ https://www.oracle.com/technetwork/tutorials/tutorials-1876574.html

Java OOA instanceof

36. What is the output of code below?

```
public class Main {
   public static void main(String[] args) {
      new Main().checkInstance(new
AtomicInteger(2));
      new Main().checkInstance(Double.valueOf(2));
      new
Main().checkInstance(Integer.valueOf(21));
   }

   public void checkInstance(Number n) {
      if (n instanceof Double d || d % 2 == 0) { // A
         System.out.print("Double :" + d);
      } else if (n instanceof Integer i && i > 10) { // B
         System.out.print("Integer :" + i);
      } else if (n instanceof AtomicInteger i && i > 10)
{ // C
         System.out.print("AtomicInteger :" + i);
      }
   }
}
```

1. code will compile
2. code will not compile due to line A
3. code will not compile due to line B
4. code will not compile due to line C

5. none of them

Correct Answer: 2,4

Explanation:

A - invalid because we used || statement. Compiler throws - java: cannot find symbol

symbol: variable d.

There can be case when variable n of type Double and is null, then line A will throw exception. So compiler already identifies it.

B - is ok

C - Invalid. On line C we compared object reference with number primitive. We need to call method get() to get value of AtomicInteger

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-instanceof-operator.html
- https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicInteger.html

Control Flow (Loop, switch, if/else)

37. Which of the following methods with switch statement will work?

```
public sealed interface SealedParent permits A, B, C
{
}
final class A implements SealedParent {}
final class B implements SealedParent {}
final class C implements SealedParent {}
----
public class Main{
   enum Color {
      WHITE, BLACK, GREEN
   }
   public static void main(String[] args) {
      new Main().checkEnum(Color.WHITE);
      new Main().checkSealed(new A());
      new Main().checkObject(Integer.valueOf(2));
   }
   public void checkSealed(SealedParent main) {
      switch (main) {
         case A o -> System.out.println("A");
         case B t -> System.out.println("B");
         case C th -> System.out.println("C");
      }
   }
   public void checkObject(Object o) {
      switch (o) {
         case Integer i ->
System.out.println("integer");
         case String s -> System.out.println("string");
         case Double d ->
System.out.println("double");
      }
   }
   public void checkEnum(Color c) {
      switch (c) {
         case WHITE -> System.out.println("WHITE");
         case BLACK -> System.out.println("BLACK");
         case GREEN -> System.out.println("GREEN");
      }
   }
}
```

1. checkSealed()
   checkEnum()
2. all methods
3. checkSealed()
4. checkEnum()
5. checkSealed()
   checkObject()

Correct Answer: 1

Explanation:

checkSealed() - will work because the compiler takes into account whether the type of a selector expression is a sealed class. The following switch expression compiles. It doesn't need a default case label because its type coverage is the classes A, B, and C, which are the only permitted subclasses of SealedParent.

checkObject() - will not compile. It needs a default statement.

checkEnum() - will compile. Because we covered all possible variants of the enum Color.

- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-switch-expressions-and-statements.html

Date/Time API

38. What is the output of code below?

```
public class Main {
   public static void main(String[] args) {
      var datetime = LocalDateTime.of(2023, 1, 30, 10,
10);
      datetime.plus(10, ChronoUnit.YEARS);
      datetime.plus(2, ChronoUnit.HOURS);
      System.out.println("year: " + datetime.getYear()
+ " month: " + datetime.getMonth() + " hours: " +
datetime.getHour());
   }
}
```

1. code will not compile
2. year: 2033 month: JANUARY hours: 12
3. year: 2033 month: FEBRUARY hours: 12
4. year: 2023 month: JANUARY hours: 10
5. year: 2023 month: FEBRUARY hours: 10

Correct Answer: 4

Explanation:

var datetime = LocalDateTime.of(2023, 1, 30, 10, 10); will output: year: 2023 month: JANUARY hours: 10

We added 10 years and 2 hours but did not reassign updated datetime with new parameters to variable datetime. That's why it will output original content.

- https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html

39. What is the output of code below?

```
public class DateSample {
   public static void main(String[] args) {
      var d = LocalDate.of(2023,1,1);
      d = d.plusYears(1);
      d = d.plusDays(2);
      d = d.plusMonths(3);
      d = d.plusHours(4);
      d = d.plusMinutes(5);
      System.out.println("updated date is : " + d);
   }
}
```

1. code will not compile
2. will throw runtime exception
3. will ouput updated date.
4. will ouput updated datetime
5. none of them

Correct Answer: 1

Explanation:

code will not compile.

LocalDate does not have methods plusHours() aand plushMinutes().

- https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html

Exception Handling

40. What is the output of code below?

```
public class Main {
   public static void main(String[] args) {
      int a = 10;
      try{
         var b = a / 0;
      } catch (Exception e) {
         switch (e) {
            case RuntimeException r ->
System.out.println("runtime");
            case IOException er ->
System.out.println("ioexception");
         }
      }
   }
}
```

1. runtime
2. ioexception
3. compile error
4. code will execute without exceptions.
5. none of them

Correct Answer: 3

Explanation:

switch pattern lack of default case that's why compile error, if there would be default case then runtime exception is thrown.

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Exception.html
- https://docs.oracle.com/en/java/javase/17/language/pattern-matching-switch-expressions-and-statements.html

41. Select statements that are true about exceptions in java.

Runtime exceptions must be wrapped with try catch statement

Error and Runtime exceptions are unchecked exceptions.

RuntimeException do not inherit the Exception class.

Try with Resource works with objects that implements AutoCloseable

try {} finally {} - will not compile

Correct Answer: 2, 4

Explanation:

1 invalid. we don't need to wrap runtime exceptions into try catch.

From definition:

RuntimeException and its subclasses are unchecked exceptions. Unchecked exceptions do not need to be declared in a method or constructor's throws clause if they can be thrown by the execution of the method or constructor and propagate outside the method or constructor boundary

2 valid. Error and Runtime are unchecked exceptions.

3 invalid. RuntimeException extends Exception.

4 valid. Try with resource works with classes that implements AutoClosable

5 invalid. actually try {} finally {} is enough to make code compile

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/RuntimeException.html
- https://www.w3schools.com/java/java_try_catch.asp
- https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html

42. Fill in the blanks.

___ - exceptions that often respond to an exception by throwing another exception.

___ - exceptions that are identified at compile time.

___ - exceptions that are thrown in an already executing program.

1. CheckedExceptions
   UncheckedExceptions
   ChainedExceptions

2. UncheckedExceptions
   ChainedExceptions
   CheckedExceptions

3. ChainedExceptions
   CheckedExceptions
   UncheckedExceptions

4. ChainedExceptions
   RuntimeExceptions
   CompileExceptions

5. none of them

Correct Answer: 3

Explanation:

CheckedExceptions - are found at compile time. Exception, FileNotFoundException etc

UncheckedExceptions - are runtime and error thrown in normally compiled and executing program.

ChainedExceptions - can throw an exception and also set initCause() which will show another exception.

- https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
- https://docs.oracle.com/javase/tutorial/essential/exceptions/chained.html

Java OOA Overloading, Overriding, Interfaces

43. Select option(s) that are valid

```
SealedParent.java
--
public sealed class SealedParent {
}
final class A extends  SealedParent{}
non-sealed class B extends  SealedParent{}
```

1. Parent.java

```
--
public class Parent {
   public final void sayHello() {
      System.out.println("parent");
   }
}

Child.java
--
public class Child extends Parent {
   @Override
   public void sayHello() {
      System.out.println("Child");
   }
}
```

2. RecParent.java

   ```
   --
   public record RecParent() {
   }

   RecChild.java
   --

   public record RecChild() extends RecParent {
   }
   ```

3.
   ```
   public abstract class AbstractClass {
   }
   ```

4. none of them

Correct Answer: 1, 4

Explanation:

1 valid. When sealed class and its children locate in same file we can omit the keyword 'permits'.

2 invalid. final method can't be overriden.

3 invalid. Record class is final by default, so it can't be extended.

4 valid. Abstract class can be empty without any abstract method.

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
- https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

44. What is the output of the code below?

```
public class A {
    public String name = "A";
    public void out() {
        System.out.print(name);
    }
}
--
public class B extends A{
    public String name = "B";

    public void out() {
        System.out.println(name);
    }
}
--
public class Main {
    public static void main(String[] args) {
        A a = new B();
        System.out.print(a.name);
        a.out();
    }
}
```

1. BA
2. AB
3. AA
4. BB
5. none of them

Correct Answer: 2

Explanation:

In java fields of the parent class are not overriden by the child class, only methods are overriden. That's why when we wrote A a = new B(). It took the field from A and calls the method out() from B.

- https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

45. What is the output of the code below?

```
public class P {
   public String name = "P";
   public P() {
      System.out.print(name + " ");
   }
}
--
public class C extends P{
   public String className = "C";
   public C() {
      System.out.println(this.className + " - " +
this.name);
   }
}
--
public class Main {
   public static void main(String[] args) {
      var c = new C();
   }
}
```

1. P C - P

2. C - null
3. code will not compile
4. C - P
5. P C - null

Correct Answer: 1

Explanation:

When we create instance of the object C then its parent class is firstly created which is P.

So the P class will output to console value "P ".

Then

will output value from the Constructor of child class C.

which is "C - P".

When in child class we use reference 'this' then it will take local if not exists then parent’s value.

- ➢ https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html

46. Select statements that are true about abstract class?

1. must have at least one abstract method
2. child is not required to implement abstract method
3. abstract class that have abstract methods can implement some of them
4. abstract class can not be sealed class.

5. abstract class can be empty without any methods

Correct Answer: 5

Explanation:

Abstract class can be empty. It is not mandatory to have the abstract method.

Child of abstract class must implement the parent's abstract method.

Abstract method must not be implemented in parent class.

Abstract class can be sealed.

- ➢ https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Arrays and Collections

47. Select option that will help to sort Persons by their age?

```
public class Main {
    public static void main(String[] args) {
        var list = new ArrayList<Person>();
                list.add(new Person(20));
                list.add(new Person(10));
                list.add(new Person(23));
                list.add(new Person(5));
        Collections.sort(list);
        list.stream().forEach(System.out::println);
    }
}
```

1.
```
public class Person {
    public int age;
    public Person(int age) {
        this.age = age; }
    @Override
    public String toString() {
        return "Person age : " + age;
    }
}
```

2.
```
public class Person {
    public int age;
    public Person(int age) {
        this.age = age;
        }
        @Override
        public int compareTo(Person p) {
            return this.age - p.age;
        }
    }

3.  public class Person implements
    Comparable<Person>{
        public int age;
        public Person(int age) {
            this.age = age; }

        @Override
        public int compareTo(Person p) {
            return this.age - p.age;  }

        @Override
        public String toString() {
            return "Person age : " + age; }
    }

4.  public class Person implements
    Comparator<Person>{
        public int age;
        public Person(int age) {
            this.age = age; }

        @Override
        public int compareTo(Person p) {
            return this.age - p.age;  }

        @Override
        public String toString() {
            return "Person age : " + age; }
    }
```

5. none of them

Correct Answer: 3

Explanation:

Collections.sort() works with objects that implements Comparable. Only option 3 is implementing that interface.

- ➢ https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html#sort(java.util.List)

48. Select option that will help to sort(ascending) List of ClassRooms by field studentCount?

```
public class ClassRoom {
    public int studentCount;
    public ClassRoom(int studentCount) {
        this.studentCount = studentCount;
    }
}
----
public class Main {
    public static void main(String[] args) {
        var list = List.of(
                new ClassRoom(21),
                new ClassRoom(34),
            new ClassRoom(10));
list.stream().sorted(getComparator()).collect(Collectors.toList());
   }
}
```

1.
```
    public static Comparator<ClassRoom> getComparator() {
        Comparator<ClassRoom> comparator = (a, b) -> {
            if(a.studentCount > b.studentCount) { return 1;}
            else if(a.studentCount == b.studentCount) {return 0;}
            else { return -1;}
        };
        return comparator;
    }
```

2.
```
    public Comparator<ClassRoom> getComparator() {
        Comparator<ClassRoom> comparator = (a, b) -> {
            if(a.studentCount > b.studentCount) { return 1;}
            else if(a.studentCount == b.studentCount) {return 0;}
            else { return -1;}
        };
        return comparator;
    }
```

3.
```
    public static Comparator<ClassRoom>
getComparator() {
        Comparator<ClassRoom> comparator = (a,
b) -> {
            if(a.studentCount > b.studentCount) {
return -1;}
            else if(a.studentCount == b.studentCount)
{return 0;}
            else { return 1;}
        };
        return comparator;
    }
```

4.
```
   public static Comparator<ClassRoom>
getComparator() {
        Comparator<ClassRoom> comparator = (a,
b) -> {
            if(a.studentCount > b.studentCount) {
return 1;}
            else if(a.studentCount == b.studentCount)
{return -1;}
            else { return 0;}
        };
        return comparator;   }
```

5. none of them

Correct Answer: 1

Explanation:

Comparator is functional interface which compare 2 items(a and b).

if a >b return 1;

else if(a<b) return -1;
else return 0;

1 valid realization of Comparator.

2 invalid. Method must be static

3. Incorrect realization of Compator logic(reversed)

4 Incorrect realization of Compator logic(if a == b return -1 is incorrect)

- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Comparator.html

49. Select statements that are true about collections in java.

A. ArrayList is synchronized and thread-safe.

B. TreeSet elements are stored in a natural order.

C. HashSet elements are sorted in a natural order

D. Vector is synchronized and thread-safe.

E. List.of() - list created using factory method can't be modified.

1. A
2. B
3. C
4. D
5. E

Correct Answer: 2, 4, 5

Explanation:

1 invalid. ArrayList is not thread-safe

2 valid. TreeSet orders items by natural order.

3 invalid. HashSet does not order items

4 valid.

5. List.of create unmodifiable collection.

- https://docs.oracle.com/javase/8/docs/api/java/util/List.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/TreeSet.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/HashSet.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Vector.html
- https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html

50. What is the output of code below?

```
public class Main {
   public static void main(String[] args) {
      List<Integer> q = new LinkedList<>();
      q.add(5);
      q.add(1);
      q.remove(0);
      q.removeIf(i -> i < 5);
      System.out.print(q);
```

```
        Queue<Integer> list = new LinkedList<>();
        list.add(1);
        list.add(3);
        list.remove(1);
        list.removeIf(i -> i < 3);
        System.out.println(list);

    }
}
```

1. [][3]
2. [][]
3. [][1]
4. [][1,3]
5. [5][3]

Correct Answer: 1

Explanation:

remove(0) of List will remove 1'st element and removeIf() will remove all items that are smaller than 5.
So at the end the list q will be empty.
remove(1) of Queue will remove element with value 1.
RemoveIf will remove items that are less than 3.
At the end queue will be [3]

- https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html
- https://docs.oracle.com/javase/8/docs/api/java/util/List.html

GROKKING THE JAVA SE 17 DEVELOPER CERTIFICATION

By going through these questions and topic you will not only expand your knowledge but also get ready for your next java certification.

Topics Covered Include:

- Arrays and Collections
- Concurrency
- Control Flow (Loop, switch, if/else)
- Date/Time API
- Exception Handling
- Garbage collection
- Java OOAinstanceof
- Java OOAOverloading, Overriding, Interfaces
- JDBC
- Lambda and Functional Interface
- Localization
- Math API
- Modules
- Arithmetic/Boolean Expressions (Operators, Promotion, Casting)
- Record
- Sealed Classes
- Streams
- Text Block
- Local Variable Type Inference
- Abstract Class