1Z0-808 PDF

Oracle 1Z0-808 PDF is available.

Enrolling now you will get access to 308 unique **1Z0-808 certification questions** at:

https://www.certification-questions.com/java8-free-pdf-download/1Z0-808-pdf.html

This is a sample demo for 1Z0-808:

Q1. Given

```
1.  class Test{
2.
3.      public static void main(String[] args){
4.
5.      int []a = {1,2,3,4,5,6};
6.      int i = a.length;
7.
8.              while(i>=1){
9.                      System.out.print(a[i]);
10.                     i--;
11.             }
12.     }
13. }
```

What would be the output, if it is executed as a program?

A. **123456**

B. **65432**

C. **12345**

D. **An exception could be thrown at runtime.**

E. **Compile error.**

D is correct.

Length of array “a” is 6, so the value of the variable i is 6. Execution of while loop will try to print array element reverse as variable “i” has initial value 6 , So trying to access element with index position 6 will cause ArrayIndexOutOfBoundsException since the array positions start with0. Hence the correct option is D.
https://www.certification-questions.com/

Exam objective: Using Loop Constructs - Create and use while loops

Creating and Using Arrays - Declare, instantiate, initialize and use a one-dimensional array

Q2. Given

```
1.  class Ex6{
2.      public static void main(String args[]){
3.          int i = 0, j=10;
4.              try{
5.                  j /=i;
6.              }
7.                  System.out.print("Divide by Zero! ");
8.              catch(Exception e){
9.                  System.out.print(“error”);
10.             }
11. }
12. }
```

What is the output?

A. **0**

B. **0 Divide by Zero!**

C. **Divide by Zero! Error**

D. **Error**

E. **Compilation fails.**

F. **An uncaught exception is thrown at runtime.**

Option E is correct.

You can’t enter code between try and catch clause. Here line 7 causes the failure. So the answer is E, if you remove line 7 then code will compile fine and provide output as error so in that case answer would be D.

Exam objective: Handling Exceptions - Create a try-catch block and determine how exceptions alter normal program flow

Q3. Consider

- **A and E are Classes**
- **B and D are interfaces**
- **C is an abstract class**

Which are true? *(Choose 3)*

A. **class F implements B ,C{ }**

B. **class F implements B{ }**

C. **class F extends A,E{ }**

D. **class F extends E{ }**

E. **class F implements B,D{ }**

B, D and E are correct.

We can use the keyword “extends” to extend either two classes or two interfaces. In class it doesn’t matter whether abstract or not. We can use the keyword “implements” to implement an interface to a class. Multiple interfaces can be implemented by class or interface **but not multiple classes.** So B,D and E are correct according to following reasons.

A is incorrect as we tried to implement an abstract class, but the abstract class should be extended. C is incorrect as we can’t extend more than one class in java.

Exam objective: Working with inheritance – implement inheritance

Q4. Given

```
1. class Test{
2.     public static void main(String[] args) {
3.         int a[] = {};
4.         System.out.print(a instanceof Object);
5.     }
6. }
```

Note: The keyword "instanceof" is use to check whether an object is of a particular type

Which is true?

A. **Will produce output as false.**
B. **Compilation fails due to error at line 3.**
C. **Will produce output as true.**
D. **Compilation fails due to error at line 4.**
E. **Length of this array is 3.**

C is correct.

Code will compile fine so C is correct. It produce true as the output as array is type of object so it returns true so the answer C is correct and A is incorrect.

Exam objective: Creating and Using Arrays - Declare instantiate, initialize and use a one-dimensional array.

Q5. Given

```
1.  class Program{
2.      static Integer i;
3.      public static void main(String [] args){
4.          try{
5.              System.out.println(i.compareTo(0));
6.          }catch ( ArithmeticException | NullPointerException e){
7.              System.out.println("Exception");
8.          }
9.      }
10. }
```

Which is the output?

A. **-1**

B. **0**

C. **1**

D. **Exception**

E. **Compilation fails.**

Option D is correct.

From java se 8, we can use catch box for multi exceptions so this code compiles fine. In given catch box it can catch both ArithmeticException and NullPointerException. At line 7, a NullPointerException will be thrown since I is not initialized. Hence option D is correct.

Exam objective : Handling Exceptions - Create a try-catch block and determine how exceptions alter normal program flow

Q6. Given

```
1.  class Ex1{
2.          public static void main(String[] args) {
3.           int a[] = { 1,2,053,4};
4.           int b[][] = { {1,2,4} , {2,2,1},{0,43,2}};
5.           System.out.print(a[3]==b[0][2] );
6.           System.out.print(" " + (a[2]==b[2][1]));
7.                   }
8.          }
```

Which is the output?

A. **true false**

B. **false false**

C. **false true**

D. **true true**

E. **Compilation fails**

D is correct.

Indexing of array elements begin with zero. So [1] refers to the second element of an array. So here a[3] refers to the fourth element of array a. It's value is 4 and we have assigned 4 to b[0][2]. We have assigned octal value to a[2] so the value of element is 43 in decimals. And we have assigned 43 in decimal to b[2][1]. So both will print true

According to above reasons A, B and C are incorrect. E is incorrect as code compiles fine.

Exam objective: Creating and Using Arrays - Declare instantiate, initialize and use a one-dimensional array.

Declare, instantiate, initialize and use multi-dimensional array

Q7. Choose three legal identifiers.

A. **2ndtName**

B. **_8_**

C. **&name**

D. **$**

E. **new**

B and D are correct.

Identifiers can start with a letter, a dollar mark or an underscore. They can have any length. It is good practice to define identifiers meaningfully not like here as it will make easy to read you code when review needed. Also we can't use any reserved keyword for variable name. Hence E is incorrect as "new" is reserved keyword.

A is incorrect as it starts with a number.

C is incorrect as it start with "**&".**

Exam objective: Working With Java Data Types - Declare and initialize variables

Q8. Given

```
1.  class Test{
2.          int value = 10;
3.      public static void main(String[] args) {
4.          new Test ().print();
5.      }
6.      public void print(){
7.          int value = 8;
8.          System.out.print(value);
9.      }
10. }
```

Which is the output?

A. 8

B. 10.

C. Compilation fails due to error at line 4.

D. Compilation fails due to error at line 7.

A is correct.

Inside a class method, when a local variable have the same name as one of the instance variable, the local variable shadows the instance variable inside the method block. So the value of “a” (8) in method print() can see as the output. So the answer is A.

B is incorrect as the instance variable is shadowed by the variable in the method print().

The code compiles fine, hence C and D are incorrect.

Exam objective: Java Basics - Define the scope of variables

Q9. Given

```
1.  class Test{
2.          static int x = 0;
3.      public static void main(String[] args) {
4.          for(int x=0;x<5;x++){    }
5.                  System.out.print(x);
6.      }
7.  }
```

Which is the output?

A. 4

B. 5

C. 0

D. Compilation fails.

E. Runtime exception will be thrown.

C is correct.

The life of Variable x ends with the end of for loop as the scope of variable x limits within for loop. So the value of static variable will print. So the output will be 0. Hence the answer is C.

A and B are incorrect as explained above.

There is no reason to fail the compilation or to throw exception, so the C and E are incorrect.

Exam objective: Java Basics - Define the scope of variables

Q10. Given

1. **class Exer{**
2. **public static void main(String [] args){**
3. **String s = "Java";**
4. **s.concat(" SE 7");**
5. **s.replaceAll("7","");**
6. **System.out.print(s);**
7. **}**
8. **}**

What is the result?

A. **Java SE ""**

B. **Java SE 7**

C. **Java SE**

D. **Java.**

E. **Compilation fails.**

D is correct.

Line three create a new String object and gives it the value "Java". Line four and five modified the String "java" but there is no reference used to refer that modified objects. So it really creates two useless String objects. So at the end s stays as "Java".

So the A, B and C are incorrect as the "String s" remains unchanged as explained above.

Exam objective: Working With Java Data Types - Create and manipulate strings

Q11. Which are true?

A. **Default constructor should be always there for any class.**

B. **Default constructor only contains "this();" .**

C. **When defining our own constructor we can't use any access modifier.**

D. **A constructor should not have a return type.**

E. **We can't change default constructor of an interface.**

D is correct.

We can use any access modifier Constructor, so C is incorrect. E is incorrect as interfaces do not have constructors.

A is incorrect as when there is user defined constructor, the default constructor vanish so it is not a must, however at least one constructor should present.

B is incorrect as default constructor has "super();".

Exam objective: Working with Methods and Encapsulation - Differentiate between default and user-defined constructors

Q12. Consider following three statements

I.Overloaded method must change the argument list.

II.Overloaded method may change the return type.

III.Overloaded method may declare broader checked exception.

Which is true?

A. **I only**
B. **II only**
C. **III only**
D. **I and II only**
E. **all**

E is correct

Answer is E correct as all given statements are correct.

Statement I is correct as overloaded method must change the argument list.

Statement II is correct as overloaded methods may change the return type.

Statement III is correct as they can throw new or broader exceptions.

Exam objective : Working with Methods and Encapsulation – Create an overloaded method

Q13. Given

```
1.  class Sup{
2.      protected void method(int x){
3.              System.out.print("Sup " + x);
4.              }
5.      }
6.      class Sub extends Sup{
7.              //overload method () here
8.      }
```

Consider

I. **public void method (){ }**

II. **private final int method (int i){ return i; }**

III. **private final void method (String s)throws Exception{ }**

Which is true?

A. **I only**

B. **II only**

C. **I and III only**

D. **I and II only**

E. **all**

C is correct.

II is incorrect since there we haven't change the argument list, which is a must for an overloading.

I and III are correct as there we have changed the argument list, overloaded method can change the return type.also can throw new or broader exceptions.

Exam objective : Working with Methods and Encapsulation – Create an overloaded method

Q14. Consider following method

```
public void method(int a){
                    }
```

Which is true?

A. **This method can only invoke through an instance of enclosing class.**
B. **This method has marked with highest restrictive access modifier.**
C. **This method returns an int.**
D. **This method takes int array as argument.**
E. **This is an incorrect method.**

A is correct

Given method is declared with public access level which is the lowest restrictive access level. Since we have used void, method can't return anything. Method takes integer as an argument. Also there we haven't use static, so it means it is an instance method, so you have to have a instance of enclosing class to access that method, hence option A is correct.

Exam objective : Working with Methods and Encapsulation – Create methods with arguments and return values.

Q15. Consider following method

```
static int min(double[] in){
    //codes
    }
```

Which is true?

A. **This method is incorrect as it doesn't have access modifier.**

B. **This method has marked with static access modifier.**

C. **This method can be used to return the minimum value of an array.**

D. **None of above.**

D is correct

A is incorrect as every method should have access level, in this case we haven't used any access modifier explicitly so it will be automatically default access level.

B is incorrect as static is not an access modifier. C is incorrect as this method take double array as the argument and return integer, since the minimum value of the array is double, we can't use this method for the task.

Exam objective : Working with Methods and Encapsulation – Create methods with arguments and return values.

Q16. Given

```
1.  class Test{
2.  public static void main(String args[]){
3.          final int j;
4.          j=2;
5.    int x= 0;
6.
7.    switch(x){
8.    case 0: {System.out.print("A");}
9.    case 1: System.out.print("B"); break;
10.   case j: System.out.print("C");
11.         }
12.   }
13. }
```

What is true?

A. **The output could be A**

B. **The output could be AB**

C. **The output could be ABC**

D. **Compilation fails.**

E. **There could be no output**

D is correct.

When using switch, case variable should be a compile time constant.

D is correct as this code fails to compile as the "case j"s "j" is not compile time constant.(Here integer j is not a compile time constant.) Other answers are incorrect as this code fails to compile.

Exam objective: Using Operators and Decision Constructs – Use switch statements

Q17. Given

```
1. class Test{
2.   public static void main(String in[]){
3.     int []in= {1,2,3};
4.       for(int x = 0;++x <4;x++)
5.         System.out.print(in[x]);
6.         }
7. }
```

What is the output?

A. **123**

B. **2 followed by an ArrayIndexOutOfBoundsException**

C. **23**

D. **Compilation fails.**

E. **No output and ArrayIndexOutOfBoundsException will be thrown at runtime**

D is correct.

It is illegal to have two variables with same name inside one scope. In this case method argument is declared with name in and also at line 3, there is another variable declared with name in, which causes a compile time error.

Exam objective : Java Basics - Define the scope of variables

Q18. Given

```
1. class Test{
2.         public static void main(String args[]){
3.                 int x = (int)args[0];
4.                 System.out.print(x);
5.         }
6. }
```

What is true?

A. **If we use command line invocation,** *java Test 5,* **the output will be 5.**

B. **If we use command line invocation,** *java Test abc,* **An ClassCastException will be thrown.**

C. **If we use command line invocation,** *java Test, a* **ArrayIndexOutOfBoundsException will be thrown.**

D. **Compilation fails due to error on line 3.**

E. **Compilation fails due to multiple errors**

D is correct.

D is correct as Java compiler already knows some inconvertible types. Here we try to convert a String to int, it cause a compile error and not ClassCastException. Code fails so E is incorrect. This code fails to compile so it won't produce any output or exception so A, B and C are incorrect. This code fails to compile so it won't produce any output or exception so A, B and C are incorrect.

Exam Objective : Working with inheritance – Determine when casting is necessary.

Handling Exceptions - Differentiate among checked exceptions, RuntimeExceptions and Errors

Q19.Which is correct?

A. **Low cohesion is better.**

B. **Loose coupling is bad.**

C. **High cohesion makes it easier to maintain program.**

D. **Tight coupling makes it easier to maintain program.**

C is correct.

Coupling is the degree that one class knows about another. As an example, if class A knows more than it should about the class B then what happens when we modify B without knowing Class A, then it may alter the Class A too. So high coupling is not good so B is incorrect.

Cohesion is used to indicate the degree to which a class has single, well focused purpose .So low cohesion class make it harder to maintain as it supports multiple unfocused roles. So C is correct and A is incorrect.

Exam Objective : Working with methods and encapsulation – Apply encapsulation principles to a class

Q20.Given

```
1.    class OCAJP {
2.        public static void main(String[] args){
3.            int x = 10;
4.            int y = x>10?1:x<10?-1:0;
5.            System.out.println(y);
6.        }
7.    }
```

What is the result?

A. 1

B. -1

C. 10

D. 0

E. Compilation fails.

D is correct.

Here we have combined two ternary operators. First one check local variable x is greater than 10 if not then it do another comparison to check if local variable x is lesser than 10, otherwise it will assign 1 to the variable y. If second condition met then it will assign -1 to the variable y, if not then 0 will be assigned to the variable y, so in this case 0 will be assigned to the variable y. Hence option D is correct.

Exam Objective : Using Operators and Decision Constructs - Create if and if/else and ternary constructs

Q21.Which of the following is correct lambda expression?

A. **Predicate<String> filter = () -> {return c.indexOf("a")>0;};**

B. **Predicate<String> filter = (c) -> return c.indexOf("a")>0;**

C. **Predicate<String> filter = (c) -> {return c.indexOf("a");};**

D. **Predicate<String> filter = (c) -> {return c.length();};**

A is correct.

A lambda expression is composed of three parts.

Argument List Arrow Token Body

The body can be either a single expression or a statement block. In the expression form, the body is simply evaluated and returned. In the block form, the body is evaluated like a method body and a return statement returns control to the caller of the anonymous method. But remember return statement is NOT an expression so in a lambda expression, you should enclose statements in braces ({...}). Also you can omit the data type of the parameters in a lambda expression.

Predicate test method returns a Boolean so C and D are incorrect.

B is incorrect as there we haven't use curly braces.

Exam Objective : Working with Selected classes from the Java API – Write a simple Lambda expression that consumes a Lambda Predicate expression

Q22. Given

```
1.  //Assume all necessary importing have done
2.
3.  public class Test{
4.
5.     public static void main(String[] args){
6.
7.          ArrayList<String> list = new ArrayList<String>();
8.          list.add("A");
9.          list.add("B");
10.         list.add("C");
11.         list.add("D");
12.
13.         Instant start = Instant.now();
14.
15.         list.forEach((s)-> System.out.println(s));
16.
17.         Instant end = Instant.now();
18.
19.         long duration = // insert here
20.    }
21. }
```

Which insert at line 19, will assign total execution time of line 13 in milliseconds to the duration variable?

A. **new Duration (start, end).getMillis());**

B. **end - start**

C. **Duration.between(start, end);**

D. **Duration.between(start, end).toMillis();**

Option D is correct.

Option D is correct as Duration measures an amount of time using time-based values. Here, to get total execution time, we cah use two java.Time.Intant instances. We can invoke the static "between" method of java.time.Duration class, which will return duration object with time difference. To convert it to milliseconds we use toMills() method.

Option A is incorrect as there is no such a constructor which takes two Instant objects for Duration class.

Option B is incorrect as we can't do such operation directly with Instant instances.

Option C is incorrect as direct result of the between method is in ISO format, not in milliseconds, for example, It would result something like "`PT0.252S`".

EXAM OBJECTIVE **:** Working with Selected classes from the Java API - Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.DateTimeFormatter, java.time.Period

Q23. Consider following interface.

```
interface Multi{
        public int multiply(int p1,int p2);
}
```

Which of the following will create instance of above interface type? *(Chose 2)*

A. **Multi mul = (x ,y) -> return x*y;**

B. **Multi mul = (x ,y) -> {return x*y;};**

C. **Multi mul = () -> {return x*y;};**

D. **Multi mul = (int x ,int y) -> {return x*y;};**

E. **Multi mul = (int x ,int y) -> return x*y;**

F. **Multi mul = () -> return x*y;**

Option B and D are correct.

Here we use lambda expression for concreting abstract method of Multi interface. A lambda expression is composed of three parts.

Argument List Arrow Token Body

The body can be either a single expression or a statement block. In the expression form, the body is simply evaluated and returned. In the block form, the body is evaluated like a method body and a return statement returns control to the caller of the anonymous method. But remember return statement is NOT an expression so in a lambda expression, you MUST enclose statements in braces ({...}). Also you can omit the data type of the parameters in a lambda expression.

As explained above, Option A and E is incorrect as there is no braces where return statement present.

Option C is incorrect as arguments not given.

Exam Objective : Working with Selected classes from the Java API – Write a simple Lambda expression that consumes a Lambda Predicate expression

Q24. Given

```
1.  import java.util. List;
2.  import java.util. ArrayList;
3.     class Test{
4.           public static void main(String [] args){
5.                 List list = new ArrayList(1);
6.                 lst.add(5);
7.                 lst.add("A");
8.                 lst.add(new Integer(5));
9.           System.out.print(lst);
10.          }
11.   }
```

Which is true?

A. The output will be [5, A, 5]

B. The output will be [ ]

C. Compilation fails due to error on line 5.

D. Compilation fails due to multiple errors.

E. An Exception is thrown at the runtime

Option A is correct.

Option A is correct as the code compiles and runs without any exception and prints [1, A, 5].

We have not specified what can List list holds, so it can take any kind of object except primitive. Hence it is completely legal to add String and Integer.

Exam Objective : Working with Selected classes from the Java API - Declare and use an ArrayList of a given type

Q25.Which of the following is a checked exception?

A. **FileNotFoundException**

B. **ArithmeticException**

C. **ClassCastException**

D. **NullPointerException**

E. **IllegalArgumentException**

A is correct.

ArithmeticException : Thrown by the JVM when code attempts to divide by zero

ClassCastException : Thrown by the JVM when an attempt is made to cast an exception to a subclass of which it is not an instance

NullPointerException Thrown by the JVM when there is a null reference where an object is required.

IllegalArgumentException : Thrown by the programmer to indicate that a method has been passed an illegal or inappropriate argument

Above all exceptions are Runtime exceptions.

FileNotFoundException : Thrown programmatically when code tries to reference a fi le that does not exist. And this is checked exception. So option A is correct.

Exam Objective : Handling Exceptions – Differentiate among checked exceptions, unchecked exceptions, and Errors