Q1. What is the size of float and double in java?

A. 32 and 64
B. 32 and 32
C. 64 and 64
D. 64 and 32

Ans. float:

- Size: 32 bits (4 bytes)
- Precision: Approximately 7 decimal digits
- Range: Approximately ±3.40282347E+38F
- Suffix: Use "f" or "F" to denote a float literal (e.g., 3.14f)

double:

- Size: 64 bits (8 bytes)
- Precision: Approximately 15 decimal digits
- Range: Approximately ±1.7976931348623157E+308
- Suffix: By default, decimal numbers with a decimal point are treated as double literals (e.g., 3.14)

Q2. Automatic type conversion is possible in which of the possible cases?

A. Byte to int

B. Int to long

C. Long to int

D. Short to int

Ans. All of the options mentioned (A. Byte to int, B. Int to long, C. Long to int, D. Short to int) are correct. Automatic type conversion is possible in all these cases:

A. Byte to int: A byte can be automatically converted to an int without any explicit casting because an int can represent a wider range of values than a byte.

B. Int to long: An int can be automatically converted to a long without any explicit casting because a long can represent a wider range of values than an int.

C. Long to int: A long can be automatically converted to an int, but there is a possibility of data loss if the long value exceeds the range of an int. In such cases, explicit casting should be used to avoid data loss.

D. Short to int: A short can be automatically converted to an int without any explicit casting because an int can represent a wider range of values than a short.

Q3.Find the output of the following code. int Integer = 24; char String = ‘I’; System.out.print(Integer); System.out.print(String);

A. Compile error

B. Throws exception

C. I

D. 24 I

Ans. `int Integer = 24;`: This line declares an integer variable named "Integer" and initializes it with the value 24.

`char String = 'I';`: This line declares a character variable named "String" and initializes it with the character 'I'.
`System.out.print(Integer);`: This line prints the value of the "Integer" variable, which is 24, to the console.
`System.out.print(String);`: This line prints the value of the "String" variable, which is 'I', to the console.
The output will be printed without any spaces between the two values, as the `System.out.print()` method does not add a newline after the printed content. Therefore, "24" and "I" will appear together in the same line.

Q4.Find the output of the following program. public class Solution{ public static void main(String[] args){ short x = 10; x = x * 5; System.out.print(x); } }

A. 50

B. 10

C. Compile error

D. Exception

Ans. `short x = 10;`: This line declares a variable "x" of type short and initializes it with the value 10, which is within the range of a short data type.

`x = x * 5;`: This line attempts to multiply the value of "x" (which is a short) by 5. The result of the multiplication will be an int because the literal "5" is of type int. In this case, Java automatically promotes the short "x" to an int, performs the multiplication, and then assigns the result back to "x".

However, the assignment statement `x = x * 5;` will result in a compilation error. The reason is that the right-hand side expression evaluates to an int, and you cannot assign an int to a short without an explicit type cast.

Q5.Find the output of the following program. public class Solution{ public static void main(String[] args){ byte x = 127; x++; x++; System.out.print(x); } }

A. -127

B. 127

C. 129

D. 2

Ans. `byte x = 127;`: This line declares a variable "x" of type byte and initializes it with the value 127. The value 127 is the maximum value that can be represented by a byte.

`x++;`: This line increments the value of "x" by 1. Since the value of "x" is 127, after this increment operation, the value will overflow because the range of a byte is from -128 to 127. When it overflows, it wraps around to the minimum value, which is -128.
`x++;`: This line increments the value of "x" by 1 again. Now "x" is -128, and after this increment operation, it will wrap around to the minimum value again, which is -128.

So, the correct option is:

A. -127

Q6. Select the valid statement.

A. char[] ch = new char(5)

B. char[] ch = new char[5]

C. char[] ch = new char()

D. char[] ch = new char[]

Ans. B. char[] ch = new char[5]

Explanation:

- A valid array declaration and initialization in Java should use square brackets "[]" after the data type to indicate that it's an array.
- The statement `char[] ch` declares an array of characters named "ch".
- The part `new char[5]` creates a new array of characters with a length of 5, meaning it can store 5 characters in total.

Q7. Find the output of the following program. public class Solution{ public static void main(String[] args){ int[] x = {120, 200, 016}; for(int i = 0; i < x.length; i++){ System.out.print(x[i] + " "); } } }

A. 120 200 016

B. 120 200 14

C. 120 200 16

D. None

Ans. In this program, we have an array of integers `x` initialized with three values: 120, 200, and 016.

Now, let's understand the values in the array:

`120`: This is a decimal literal, and it is represented as-is.

`200`: This is also a decimal literal, and it is represented as-is.

`016`: This is an octal literal. In Java, if a numeric literal starts with a leading '0', it is treated as an octal number. The decimal value of 016 in octal is 1 * 8^1 + 6 * 8^0 = 8 + 6 = 14.

So, the correct option is:

B. 120 200 14

Q8.When an array is passed to a method, what does the method receive?

A. The reference of the array

B. A copy of the array

C. Length of the array

D. Copy of first element

Ans. A. The reference of the array

When an array is passed to a method in Java, the method receives the reference (or memory address) of the array and not a copy of the array itself. Arrays are reference types in Java, and they store the memory address where the elements of the array are located in the memory.

When you pass an array as an argument to a method, you are essentially passing the memory address of the array. This means that any changes made to the array elements

inside the method will affect the original array outside the method as well because they both point to the same memory location.

Q9.Find the value of A[1] after execution of the following program. int[] A = {0,2,4,1,3}; for(int i = 0; i < a.length; i++){ a[i] = a[(a[i] + 3) % a.length]; }

A. 0

B. 1

C. 2

D. 3

Ans. In this program, we have an array `A` with elements: {0, 2, 4, 1, 3}.

Now, let's understand the logic inside the for loop:

`A[i] + 3`: The value of the current element at index `i` is retrieved (let's say it's `x`), and 3 is added to it.

`(A[i] + 3) % A.length`: The result of step 1 is then taken modulo (`%`) the length of the array `A`. This ensures that the index wraps around within the bounds of the array.

For `i = 0`:

- `A[0]` is 0, so `x = 0 + 3 = 3`.
- `3 % 5` is 3 (since the length of `A` is 5).
- So, `A[0]` is set to `A[3]`, which is 1.

For `i = 1`:

- `A[1]` is 1, so `x = 1 + 3 = 4`.

- `4 % 5` is 4.
- So, `A[1]` is set to `A[4]`, which is 3.

For `i = 2`:

- `A[2]` is 4, so `x = 4 + 3 = 7`.
- `7 % 5` is 2.
- So, `A[2]` is set to `A[2]`, which is 4 (no change).

For `i = 3`:

- `A[3]` is 1, so `x = 1 + 3 = 4`.
- `4 % 5` is 4.
- So, `A[3]` is set to `A[4]`, which is 3.

For `i = 4`:

- `A[4]` is 3, so `x = 3 + 3 = 6`.
- `6 % 5` is 1.
- So, `A[4]` is set to `A[1]`, which is 2.

After executing the for loop, the updated array `A` will be: {1, 3, 4, 3, 2}.

So, the value of `A[1]` after the execution of the program is:

B. 3

Q10. When is the object created with a new keyword?

A. At run time

B. At compile time

C. Depends on the code

D. None

Ans. A. At run time

In Java, objects are created dynamically at runtime using the "new" keyword. When you use the "new" keyword to create an object of a class, memory is allocated to store the object, and the constructor of the class is called to initialize the object's state. This process happens at runtime when the program is executing.

Q11. Identify the corrected definition of a package.

A. A package is a collection of editing tools

B. A package is a collection of classes

C. A package is a collection of classes and interfaces

D. A package is a collection of interfaces

Ans. C. A package is a collection of classes and interfaces

The correct definition of a package in Java is option C. A package is a way to organize and group related classes and interfaces together. It provides a namespace for the classes, preventing naming conflicts and making it easier to manage the code.

A Java package is a directory that contains a set of Java class files. These class files can include regular classes as well as interfaces. By organizing classes and interfaces into packages, Java developers can create a modular and maintainable code structure, making it easier to develop, reuse, and manage their codebase.

Q12.Identify the keyword among the following that makes a variable belong to a class,rather than being defined for each instance of the class.

A. final

B. static

C. volatile
D. abstract

Ans. B. static

The keyword "static" makes a variable belong to a class rather than being defined for each instance (object) of the class. When you declare a variable as static, it is associated with the class itself rather than with individual instances of the class.

Q13.Identify what can directly access and change the value of the variable res. Package com.mypackage; Public class Solution{ Private int res = 100; }
A. Any class
B. Only Solution class
C. Any class that extends Solution
D. None

Ans. B. Only Solution class

In the given code snippet, the variable "res" is declared as private within the class "Solution." The "private" access modifier restricts direct access to the variable "res" from outside the class.

Q14.In which of the following is the toString() method defined?
A. java.lang.Object
B. java.lang.String
C. java.lang.util
D. None

Ans. A. java.lang.Object

The `toString()` method is defined in the `java.lang.Object` class, which is the root class for all Java classes. Every class in Java implicitly extends the `Object` class, directly or indirectly. As a result, all Java objects inherit the `toString()` method from `Object`.

The `toString()` method in the `Object` class is intended to provide a string representation of the object. It is often used for debugging or displaying useful information about the object.

If a class does not override the `toString()` method, it will inherit the default implementation from `Object`, which returns a string containing the class name and the object's hash code.

Q15.Identify the output of the following program. String str = “abcde”;
System.out.println(str.substring(1, 3));
A. abc
B. bc
C. bcd
D. cd

Ans. `str.substring(1, 3)`:

- `beginIndex` is 1 (inclusive), which corresponds to the character 'b'.
- `endIndex` is 3 (exclusive), which corresponds to the character 'd'.
- The substring will include the characters at index 1 and 2, i.e., 'b' and 'c'.
- The character at index 3 ('d') is not included in the substring.

So, the correct answer is:

B. bc

Q16.Identify the output of the following program. String str = “Hellow”;
System.out.println(str.indexOf(‘t));
A. 0
B. 1
C. true
D. -1

Ans. The `indexOf(char ch)` method in Java returns the index of the first occurrence of the specified character `ch` in the given string. If the character is not found in the string, it returns -1.

In the original string "Hellow", there is no occurrence of the character 't', so the `indexOf('t')` method will return -1.

So, the correct answer is:

D. -1

Q17. Identify the output of the following program. Public class Test{ Public static void main(String argos[]){ String str1 = “one”; String str2 = “two”; System.out.println(str1.concat(str2)); } }

A. one
B. two
C. onetwo
D. twoone

Ans. The `concat()` method in Java is used to concatenate two strings together. It returns a new string that is the concatenation of the calling string (in this case, `str1`) followed by the string argument (in this case, `str2`).

In this program, the value of `str1` is "one" and the value of `str2` is "two". When `str1.concat(str2)` is executed, it will concatenate `str2` at the end of `str1`, resulting in a new string "onetwo".

So, the correct answer is:

C. onetwo

Q18. How many objects will be created in the following? String a = new String(“FlipRobo”); String b = new String(“FlipRobo”); String c = “FlipRobo”; String d = “FlipRobo”;

A. 2

B. 3

C. 4

D. None

Ans. `String a = new String("FlipRobo");`: This line explicitly creates a new object of type `String` using the `new` keyword and assigns it to the reference variable `a`. Since the `new` keyword is used, a new `String` object is created in the heap memory. This creates one object.

`String b = new String("FlipRobo");`: This line is similar to the previous line. It also explicitly creates a new object of type `String` using the `new` keyword and

assigns it to the reference variable `b`. Again, this creates another new `String` object in the heap memory. This creates one more object.
`String c = "FlipRobo";`: This line creates a new `String` object in the string pool and assigns it to the reference variable `c`. Since the literal "FlipRobo" is used and it is the same as the value of the string assigned to `a` and `b`, Java optimizes by reusing the existing "FlipRobo" string object from the string pool. So, no new object is created here. This does not create any additional objects.`String d = "FlipRobo";`: This line also creates a new `String` object in the string pool and assigns it to the reference variable `d`. As with the previous line, Java optimizes by reusing the existing "FlipRobo" string object from the string pool. No new object is created here either. This does not create any additional objects.

So, the total number of objects created is:

1 (for `a`) + 1 (for `b`) + 0 (for `c`) + 0 (for `d`) = 2

The correct answer is:

A. 2

Q19.Find the output of the following code. int ++a = 100; System.out.println(++a);

A. 101

B. Compile error as ++a is not valid identifier

C. 100

D. None

Ans. The code you provided has a compilation error. The statement `int ++a = 100;` is not valid because you cannot use the increment operator (`++`) while declaring a variable. The increment operator is used to modify the value of a variable after its declaration, not during the declaration itself.

Q20.Find the output of the following code. if(1 + 1 + 1 + 1 + 1 == 5){ System.out.print(“TRUE”); } else{ System.out.print(“FALSE”); }

A. TRUE

B. FALSE

C. Compile error

D. None

Ans. The expression `1 + 1 + 1 + 1 + 1` evaluates to `5`, and the condition `5 == 5` is true. Therefore, the code inside the if-block will be executed.

So, the correct answer is:

A. TRUE

Q21.Find the output of the following code. Public class Solution{ Public static void main(String args[]){ Int x = 5; x * = (3 + 7); System.out.println(x);

A. 50

B. 22

C. 10

D. None

Ans. `int x = 5;`: This line declares an integer variable `x` and initializes it with the value `5`.

`x *= (3 + 7);`: This line multiplies the current value of `x` (which is `5`) by the result of `(3 + 7)`, which is `10`. It is equivalent to `x = x * 10`.
`System.out.println(x);`: This line prints the value of `x` to the console.

The value of `x` after the operation is `5 * 10`, which is `50`. Therefore, the output of the code will be:

So, the correct answer is:

A. 50

Q22.Identify the return type of a method that does not return any value.

A. int

B. void

C. double

D. None

Ans. B. void

The return type of a method that does not return any value is `void`. In Java, the keyword `void` is used to indicate that a method does not return any value. When a method is declared with a return type of `void`, it means that the method performs certain actions or operations but does not provide any result back to the caller.

Q23.Output of Math.floor(3.6)?

A. 3

B. 3.0

C. 4

D. 4.0

Ans. The `Math.floor()` method in Java returns the largest integer value that is less than or equal to the given argument. In other words, it rounds down the decimal value to the nearest integer value. Let's find the output of `Math.floor(3.6)`:

The given argument is `3.6`. The largest integer value less than or equal to `3.6` is `3`.

So, the output of `Math.floor(3.6)` will be:

A. 3

Q24. Identify the modifier which cannot be used for constructor.

A. public

B. protected

C. private

D. static

Ans. D. static

The "static" modifier cannot be used for constructors in Java. Constructors are special methods used to initialize objects of a class, and they are not associated with any specific instance or object. They are called automatically when an object is created using the "new" keyword.

Q25. What are the variables declared in a class for the use of all methods of the class called?

A. Object

B. Instance variables

C. Reference variable

D. None

Ans. B. Instance variables

Variables declared in a class that are associated with the instances (objects) of the class are called "instance variables." These variables hold data that is unique to each instance of the class. They are also known as "member variables" or "attributes."

Q26. Find the output of the following code. Public class Solution{ Public static void main(String args[]){ Int i; for(i = 1; i < 6; i++){ if(i > 3) continue; } System.out.println(i); } }

A. 3

B. 4

C. 5

D. 6

Ans. In the given code, the `for` loop runs from `i = 1` to `i < 6`, incrementing `i` by 1 each iteration. Inside the loop, there is an `if` statement with the condition `i > 3`. If `i` is greater than 3, the `continue` statement will be executed, which means the loop will skip the rest of the current iteration and move on to the next iteration.

So, the correct answer is:

D. 6

Q27. Identify the infinite loop.

A. for(; ;)

B. for(int i = 0; i < 1; i--)

C. for(int i = 0; ;i++)

D. All of the above

Ans. D. All of the above

All three options (A, B, and C) represent different ways to create infinite loops:

A. `for( ; ; )`: This is a standard infinite loop in Java. It has no initialization (`for( ;`), `no termination condition (;), and no update statement ()`). Since there is no termination condition, the loop will run indefinitely.

B. `for(int i = 0; i < 1; i--)`: This loop also creates an infinite loop. The loop starts with `i = 0`, but the termination condition `i < 1` is always true because `0` is less than `1`. Additionally, the update statement `i--` decrements the value of `i` with each iteration, but it doesn't change the fact that `i` will always be less than `1`.

C. `for(int i = 0; ; i++)`: This loop is also an infinite loop. The loop starts with `i = 0`, and there is no termination condition (`;`). Without a termination condition, the loop will keep running indefinitely.

Q28.Exception created by try block is caught in which block

A. catch

B. throw

C. final

D. none

Ans. A. catch

In Java, when an exception is thrown within a try block, it can be caught and handled in a catch block. The try-catch mechanism allows you to write code that can gracefully handle exceptional situations without terminating the program abruptly.

Q29.Which of the following exception is thrown when divided by zero statement is executed?

A. NullPointerException

B. NumberFormatException

C. ArithmeticException

D. None

Ans. C. ArithmeticException

The `ArithmeticException` is thrown when an arithmetic operation (such as division or modulo) is performed with inappropriate operands. One of the common scenarios that triggers an `ArithmeticException` is dividing a number by zero, which is mathematically undefined.

Q30.Where is System class defined?

A. java.lang.package

B. java.util.package

C. java.io.package

D. None

Ans. A. java.lang.package

The `System` class is defined in the `java.lang` package in Java. The `java.lang` package is a special package that is automatically imported into all Java programs. It contains fundamental classes and provides the basic functionalities required by all Java programs.

The `System` class contains several useful methods and fields related to the Java runtime system, environment, and I/O operations. Some of the commonly used methods in the `System` class include `out.println()`, `in`, and `currentTimeMillis()`.