# JAVA QUESTION BANK

**Q.1 What is oops ?**

**Answer:** OOPs stands for Object-Oriented Programming. It's a programming paradigm based on the principles of encapsulation, inheritance, polymorphism, and abstraction. It organizes code into objects that contain data and methods to operate on that data, promoting code reusability and modularity.

**Q.2 List out features of oops**

Answer: Features of oops are as follow:

A. Object
B. Classes
C. Inheritance
D. Encapsulation
E. Polymorphism
F. Data Abstraction
G. Association
H. Method Overriding
I. Method Overloading
J. Constructor & Destructors

**Q.3 Give the ex of objects & list out its properties & characteristics**

Answer: A simple example of a "Car" object in an Object-Oriented Programming context.

**Object: Car**
**Properties:**

1. **Make**: The brand or manufacturer of the car (e.g., Toyota, Ford).
2. **Model**: The specific model of the car (e.g., Camry, Mustang).
3. **Year**: The manufacturing year of the car.
4. **Color**: The color of the car.
5. **Mileage**: The total distance the car has traveled.
6. **EngineSize**: The size of the car's engine in litres.
7. **FuelType**: The type of fuel the car uses (e.g., gasoline, diesel).
8. **Price**: The current market price of the car.

**Characteristics:**

1. **Encapsulation**: The properties of the car object (Make, Model, Year, etc.) are encapsulated within the object, meaning they are grouped together and accessed through methods.
2. **Abstraction**: We can create methods to operate on the car object (e.g., start(), accelerate(), brake()) without needing to know the internal details of how these operations are implemented.

**Q.4 List the features of java programming languages**

**Answer: Features of java programming languages are as follow:**

1. **Platform-Independent**: Java code is compiled into bytecode, which can run on any Java Virtual Machine (JVM), making Java platform-independent.
2. **Object-Oriented:** Java supports the principles of Object-Oriented Programming (OOP), including encapsulation, inheritance, polymorphism, and abstraction.
3. **Robust and Secure:** Java incorporates strong memory management, automatic garbage collection, and exception handling to provide robustness. It also includes features like bytecode verification to enhance security.

4. **High Performance:** With the Just-In-Time (JIT) compiler, Java achieves high performance by converting bytecode to native machine code at runtime.
5. **Dynamic:** Java supports dynamic loading of classes and dynamic memory allocation, providing greater flexibility and adaptability.

**Q.5 List out the historical time span for java**

Answer: A concise overview of the historical time span for Java:
**1991:** Java's initial development began at Sun Microsystems.
**1995:** Java 1.0 was publicly released.
**1996-2000:** Introduction of Java 1.1, J2SE 1.2, J2EE 1.2, and J2ME.
**2002:** Release of J2SE 1.4.
**2004:** Java 5 introduced major language enhancements.
**2006:** Release of Java 6 with scripting support and JVM improvements.
**2011:** Oracle acquired Sun Microsystems.
**2014:** Java 8 introduced lambda expressions and the Stream API.
**2017:** Java 9 introduced the module system.
**2018:** Releases of Java 10 and Java 11 with local-variable type inference and HTTP client.
**2019:** Releases of Java 12 and Java 13 with switch expressions and Shenandoah garbage collector.
**2020:** Releases of Java 14 and Java 15 with pattern matching and Records.
**2021:** Releases of Java 16 and Java 17 with sealed classes and preview features.

**Q.6 What is type of casting in java ?**

Answer: **Implicit Casting:**
- This type of casting is done automatically by the compiler.
- It occurs when a smaller data type is converted to a larger data type.
- No explicit casting operator is required.

Example:
```
int num1 = 10;
double num2 = num1; // Implicit casting from int to double
```
Explicit Casting:
- This type of casting is done manually by the programmer.
- It occurs when a larger data type is converted to a smaller data type.
- An explicit casting operator is required.
- There is a risk of data loss as the larger data type may not fit into the smaller data type.

Example:
```
double num1 = 10.5;
int num2 = (int) num1;  // Explicit casting from double to int
```

**Q.7 Give example of Implicit and Explicit type casting**

Answer: **Implicit Casting**: Automatic conversion from smaller to larger data types.
Byte > Short > Char > Int > Long > Float > Double
```
Example: int a=10
         Float b=a;
             b= 10.00
```
**Explicit Casting**: Manual conversion from larger to smaller data types using the casting operator **(dataType)**.
Double > Float > Long > Int > Char > Short > Byte
Example: double d=10.15

```
        Int a=(int)d;
            a=10
```

**Q.8 What is the use of break and continue keywords**

Answer: **break**: The **break** keyword is used to exit or terminate a loop prematurely. When a **break** statement is encountered inside a loop, the loop is immediately terminated, and the program control resumes at the next statement following the loop.

**Example:**

```
for(int i = 1; i <= 10; i++) {
   if(i == 5) {
      break;  // Terminate the loop when i is 5
   }
   System.out.println(i);
}
```

**break**: Exits the loop prematurely.

**continue**: Skips the current iteration and continues with the next iteration of the loop.

**Output:**

```
1
2
3
4
```

**Continue**: The **continue** keyword is used to skip the current iteration of a loop and continue with the next iteration. When a **continue** statement is encountered inside a loop, the current iteration of the loop is terminated, and the program control jumps to the beginning of the next iteration.

Example:

```
for(int i = 1; i <= 10; i++) {
   if(i == 5) {
      continue;  // Skip the iteration when i is 5
   }
   System.out.println(i);
}
```

Output:

```
1
2
3
4
6
7
8
9
10
```

**Q.9 List the keywords in java**

Answer:

1. int
2. char
3. float

4. if
5. else
6. long
7. double
8. short
9. Boolean
10. Byte
11. Class
12. break
13. continue
14. void
15. switch
16. while
17. do
18. return
19. default
20. public

Q.10 Write a java program to find out greatest number between two numbers

**Program:**

```
public class GreatestNumber {
   public static void main(String[] args) {
       int num1 = 25;
       int num2 = 45;

       if(num1 > num2) {
           System.out.println("The greatest number is: " + num1);
       } else {
           System.out.println("The greatest number is: " + num2);
       }
   }
}
```

**Output:**

```
The greatest number is: 45
```

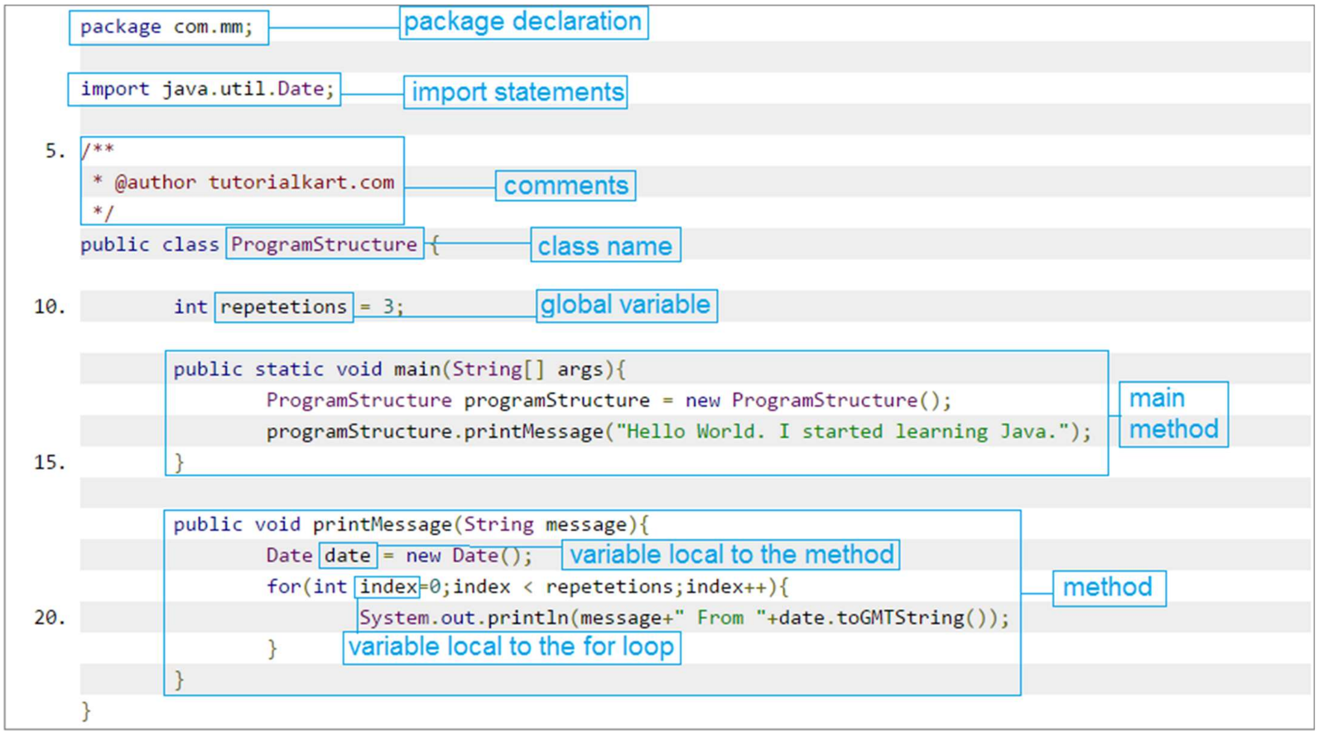
Q.1 Give different between C++ and Java

Answer:

| **C++** | **Java** |
|---|---|
| C++ is a platform dependent language. | Java is a platform independent language. |
| C++ also called semi-oops language. | Java is called complete oops language. |
| C++ support multiple inheritance. | Java doesn't support multiple inheritance. |
| C++ support constructor and destructor. | Java supports only constructor. |
| C++ support three access specifiers like-(Public, Protected, Private) | Java supports four access modifiers like-(Public, Protected, Private and default) |
| C++ support (::) Operator and operator overloading. | Java doesn't support (::) as well as operator overloading. |
| C++ support pointer. | Java doesn't support pointers. |
| C++ support pre-processor | Java doesn't support pre-processor |
| In C++ there are three types | In java support four types of loop |

Q.2 Explain the structure of java program.

Answer:



```
package com.mm;

import java.util.Date;

/**
 * @author tutorialkart.com
 */
public class ProgramStructure {

        int repetetions = 3;

        public static void main(String[] args){
                ProgramStructure programStructure = new ProgramStructure();
                programStructure.printMessage("Hello World. I started learning Java.");
        }

        public void printMessage(String message){
                Date date = new Date();
                for(int index=0;index < repetetions;index++){
                        System.out.println(message+" From "+date.toGMTString());
                }
        }
}
```

Q.3 Wap to find even odd no. between 1 to 100

Answer:

```
public class EvenOddNumbers {
   public static void main(String[] args) {
       // Find and print even numbers between 1 and 100
       System.out.println("Even numbers between 1 and 100:");
       for (int i = 1; i <= 100; i++) {
           if (i % 2 == 0) {
               System.out.print(i + " ");
           }
```

```
        }
        System.out.println();  // Print a new line

        // Find and print odd numbers between 1 and 100
        System.out.println("Odd numbers between 1 and 100:");
        for (int i = 1; i <= 100; i++) {
            if (i % 2 != 0) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output:

```
Even numbers between 1 and 100:
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
74 76 78 80 82 84 86 88 90 92 94 96 98 100

Odd numbers between 1 and 100:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73
75 77 79 81 83 85 87 89 91 93 95 97 99
```

Q.4 Wap to find prime no. between 1 to 100.

Answer:

```
public class PrimeNumbers {
    public static void main(String[] args) {
        System.out.println("Prime numbers between 1 and 100:");

        for (int i = 2; i <= 100; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
            }
        }
    }
    public static boolean isPrime(int num) {
        if (num <= 1) return false;
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) return false;
        }
        return true;
    }
}
```

Output:

```
Prime numbers between 1 and 100:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```