1. what is metaspace and heap memory

Ans: **Metaspace** is a new memory space – starting from the Java 8 version; it has replaced the older PermGen memory space. The most significant difference is how it handles memory allocation. Specifically, this native memory region grows automatically by default.

Heap memory, also known as "dynamic" memory, is an alternative to local stack memory. Local memory is quite automatic. Local variables are allocated automatically when a function is called, and they are deallocated automatically when the function exits.

2. generate multiples of 2 until 20 using recursive function

Ans:
```
package com.pack;

import java.util.Scanner;

public class tableoftwo {

	static void MultiplicationTable(int number, int i) {
		System.out.println(number + "*" + i + "=" + number * i);
		if (i < 10) {
			MultiplicationTable(number, i + 1);
		}
	}

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

		int number;
		System.out.println("Enter a number: ");
		number = sc.nextInt();
		System.out.println("Multiplication Table of " + number + " is: ");
		MultiplicationTable(number, 1);
		sc.close();
	}
}
```

3. check if two strings are equal or not

Ans:
```
package com.pack;

import java.util.Scanner;

public class twostring {
	public static void main(String args[]) {
		String str1, str2;
		Scanner scanner = new Scanner(System.in);

		System.out.println("Enter first String");
		str1 = scanner.nextLine();

		System.out.println("Enter second String");
		str2 = scanner.nextLine();

		if (str1.equals(str2))
			System.out.print("Equal Strings");
		else
			System.out.print("UnEqual Strings");
```

```
    }
}
```

4. print the character count in a string say

string s ="helloworld" print h-1, e-1, l-3,o-2

Ans:
```
package com.pack;

public class CharCount {

	public static void main(String[] args) {
		String str = "helloworld";

		int count[] = new int[256];

		int len = str.length();

		for (int i = 0; i < len; i++) {
			count[str.charAt(i)]++;
		}
		char ch[] = new char[str.length()];
		for (int i = 0; i < len; i++) {
			ch[i] = str.charAt(i);
			int find = 0;
			for (int j = 0; j <= i; j++) {
				if (str.charAt(i) == ch[j]) {
					find++;
				}
			}
			if (find == 1) {
				System.out.println(str.charAt(i) + " - " +
count[str.charAt(i)]);
			}
		}
	}
}
```

5. why java is platform independent

Ans: Java is platform-independent because it does not depend on any type of platform. Hence, Java is platform-independent language. In Java, programs are compiled into byte code and that byte code is platform-independent. ... Any machine to execute the byte code needs the Java Virtual Machine.

6. can we create class as final

Ans: You can declare some or all of a class's methods final. You use the final keyword in a method declaration to indicate that the method cannot be overridden by subclasses. The Object class does this—a number of its methods are final .

The main purpose of using a class being declared as final is to prevent the class from being subclassed. If a class is marked as final then no class can inherit any feature from the final class. You cannot extend a final class. If you try it gives you a compile time error.

7. considder we have employee class with empid, empname and salary

and list of employees get the the highest salary paid employee data

Ans:

```
package com.pack;
import java.util.Arrays;
import java.util.List;

public class Employee {
	private long id;
	private String name;
	private int salary;

	public Employee(long id, String name, int salary) {
		this.id = id;
		this.name = name;
		this.salary = salary;
	}
	public int getSalary() {
		return salary;
	}
	public void setSalary(int salary) {
		this.salary = salary;
	}
	@Override
	public String toString() {
		return "Employee [id=" + id + "," + " name=" + name + "," + " salary=" + salary + "]";
	}
}
class FindEmployee {
	public static void main(String[] args) {
		List<Employee> employees = Arrays.asList(new Employee(101, "Jay", 5000), new Employee(109, "Atul",
3000),
					new Employee(111, "Sourav", 4400));

		int maxSalary = employees.stream().map(Employee::getSalary).max(Integer::compare).get();
```

```
                System.out.println("Max salary of the employee:" + maxSalary);

                System.out.print("Employee details:");


                employees.stream().filter(emp -> emp.getSalary() == maxSalary).forEach(System.out::println);


        }


    }
```

8. consider a list of duplicate values remove duplicate value and get unique values from the list

Ans:

```
package com.pack;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Listp {

        public static void main(String[] args) {
            List<Integer> lst = new ArrayList<>();
            lst.add(1);
            lst.add(3);
            lst.add(5);
            lst.add(3);
            lst.add(5);
            lst.add(4);
            List<Integer> dlst =
lst.stream().distinct().collect(Collectors.toList());
            System.out.println(dlst);
        }
    }
```

9. can we write try and finally without catch block what is the use

Ans: Yes, we can have try without catch block by using finally block. You can use try with finally. As you know finally block always executes even if you have exception or return statement in try block except in case of System.

10. Welcome to College of Management

Ans:

```
package com.pack;

import java.util.Scanner;

public class Application {

        public static void main(String[] args) {
                System.out.println("Welcome to College of Management");
                try (Scanner sc = new Scanner(System.in)) {
                        int studentId;
                        boolean n = true;
                        String name = "Aditya";
                        int hindi;
                        int english;
```

```
                int maths;
                int science;
                int social;
                int total;
                double percentage;
                while (n) {
                        System.out.println("Do you want to continue ");
                        char go = sc.next().charAt(0);
                        if (go == 'y') {
                                System.out.println("enter valid choice");
                                System.out.println("c - check student result");
                                System.out.println("a - add student result");
                                System.out.println("x - exit");
                                char choice = sc.next().charAt(0);
                                switch (choice) {
                                case 'a':
                                        System.out.println("Add student result");
                                        System.out.println("StudentId-");
                                        studentId = sc.nextInt();
                                        System.out.println("marks in hindi-");
                                        hindi = sc.nextInt();
                                        System.out.println("marks in english-");
                                        english = sc.nextInt();
                                        System.out.println("marks in maths-");
                                        maths = sc.nextInt();
                                        System.out.println("marks in science-");
                                        science = sc.nextInt();
                                        System.out.println("marks in social-");
                                        social = sc.nextInt();
                                        System.out.println("Student added successfully");

                                        break;
                                case 'c':
                                        System.out.println("Check student result");
                                        System.out.println("Enter Student ID=");
                                        studentId = sc.nextInt();
                                        if (studentId != 38) {
                                                System.out.println("Student id not found");
                                        } else {
                                                hindi = 50;
                                                english = 90;
                                                maths = 60;
                                                science = 40;
                                                social = 85;
                                                total = hindi + english + maths + science + social;
                                                percentage = (total * 100) / 500.00;
                                                System.out.println("Student Result-");
                                                System.out.println("id-" + studentId);
                                                System.out.println("studentname-" + name);
                                                System.out.println("marks in hindi-" + hindi);
                                                System.out.println("marks in english-" + english);
                                                System.out.println("marks in maths-" + maths);
                                                System.out.println("marks in science-" + science);
                                                System.out.println("marks in social-" + social);
                                                if (percentage >= 50)
                                                        System.out.println("result-pass");
                                                else
                                                        System.out.println("result-fail");
                                                System.out.println("total-" + total);
                                                System.out.println("percentage-" + percentage);
                                        }
                                        break;
                                case 'x':
                                        return;
                                default:
                                        System.out.println("Invalid Input");

                                }
                        } else {
                                System.out.println("bye");
                                break;
                        }
                }
        }

    }
}
```

11. what is garbage collector and how it works

Ans.

- Garbage collection in Java is the process by which Java programs perform automatic memory management.
- Java programs compile to bytecode that can be run on a Java Virtual Machine.
- The garbage collector finds these unused objects and deletes them to free up memory.
- All objects are allocated on the heap area managed by the JVM. ... As long as an object is being referenced, the JVM considers it alive.
- Once an object is no longer referenced and therefore is not reachable by the application code, the garbage collector removes it and reclaims the unused memory.

12. what is heap space

Ans:

The Java heap is the area of memory used to store objects instantiated by applications running on the JVM. When the JVM is started, heap memory is created and any objects in the heap can be shared between threads as long as the application is running. The size of the heap can vary, so many users restrict the Java heap size to 2-8 GB in order to minimize garbage collection pauses.

13. what is java memory model

Ans:

The Java memory model specifies how the Java virtual machine works with the computer's memory (RAM). ... The Java memory model specifies how and when different threads can see values written to shared variables by other threads, and how to synchronize access to shared variables when necessary.In computing, a memory model describes the interactions of threads through memory and their shared use of the data.

14. what is young and old generations

Ans: **Young Generation** is where all new objects are allocated and aged. When the young generation fills up, this causes a minor garbage collection. A young generation full of dead objects is collected very quickly. Some survived objects are aged and eventually move to the old generation.

Old Generation is used to store long surviving objects. Typically, a threshold is set for young generation object and when that age is met, the object gets moved to the old generation. Eventually the old generation needs to be collected. This event is called a major garbage collection.

15. what is eden and survivor space

Ans: In Young Generation It is a place where an object lived for a short period and it is divided into two spaces:

Eden Space: The pool from which memory is initially allocated for most objects.

Survivor Space: The pool containing objects that have survived the garbage collection of the Eden space.

16. JdbcDemo database

Ans:

100 sham 100

200 ram 2000

400 vinod 5000

300 abhi 10

```
package x.pack;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Scanner;

public class JdbcDemo {

	public static void main(String[] args) throws SQLException, ClassNotFoundException {
		// TODO Auto-generated method stub
		Class.forName("org.apache.derby.client.ClientAutoloadedDriver");// loading drivers
		Connection conn = DriverManager.getConnection("jdbc:derby://localhost:1527/training;create=true",
"derby",
					"derby");
		@SuppressWarnings("resource")
		Scanner sc = new Scanner(System.in);
		char choice;
```

```
		do {
			System.out.println("1.View Employee Data");
			System.out.println("2.Update Employee Data");
			System.out.println("3.Add Employee ");
			System.out.println("4.Delete Employee Data");
			System.out.println("5.View All Employyee Data");
			System.out.println("6.Exit");
			int n = sc.nextInt();
			switch (n) {
			case 1:
				PreparedStatement st = conn.prepareStatement("select * from app.employee
where emp_id=?");
				System.out.println("Enter employee id to search data: ");
				int ie = sc.nextInt();
				st.setInt(1, ie);
				ResultSet rs = st.executeQuery();
				while (rs.next()) {
					System.out.println(
							"ID: " + rs.getString(1) + " Name : " + rs.getString(2) + "
Salary : " + rs.getString(3));
				}
				break;
			case 2:
				System.out.println("Enter employee id where you want to update his data: ");
				int iu = sc.nextInt();
				System.out.println("Enter name: ");
				String en = sc.next();
				System.out.println("Enter Salary: ");
				int esal = sc.nextInt();
				PreparedStatement st1 = conn
						.prepareStatement("UPDATE app.employee SET emp_nm = ?,
emp_sal = ? WHERE emp_id=?");
				st1.setString(1, en);
				st1.setInt(2, esal);
				st1.setInt(3, iu);
				st1.executeUpdate();
```

```
                System.out.println("Updated Successfully");
                break;
            case 3:
                PreparedStatement st2 = conn.prepareStatement("insert into app.employee
values(?,?,?)");
                System.out.println("ENTER EMPLOYEE DATA ");
                System.out.println("Employee Id= ");
                int addid = sc.nextInt();
                System.out.println("Employee Name= ");
                String addname = sc.next();
                System.out.println("Employee Salary= ");
                int addsalary = sc.nextInt();
                st2.setInt(1, addid);
                st2.setString(2, addname);
                st2.setInt(3, addsalary);
                st2.executeUpdate();
                System.out.println("Employee Data Added Successfully");
                break;

            case 4:
                PreparedStatement st3 = conn.prepareStatement("Delete from app.employee
where emp_id=?");
                System.out.println("Enter employee id to delete his data: ");
                int dd = sc.nextInt();
                st3.setInt(1, dd);
                st3.executeUpdate();
                System.out.println("Data Deleted Successfully");
                break;
            case 5:
                PreparedStatement st4 = conn.prepareStatement("select * from app.employee");
                ResultSet rs4 = st4.executeQuery();
                while (rs4.next()) {
                    System.out.println(
                            "ID: " + rs4.getString(1) + " Name: " + rs4.getString(2)
+ " Salary: " + rs4.getString(3));
                }
```

```
                    break;
                case 6:
                    System.exit(0);
                default:
                    System.out.println("Invalid Choice!!!");
                    break;
                }
                System.out.println("Do you want to continue (Y/N)");
                choice = sc.next().charAt(0);
            } while (choice == 'Y' || choice == 'y');
            System.out.println("Bye");
        }

}
```