Core Java Interview Questions (Only Syntax based)

1. Write a Code for Single Inheritance

```java
public class Employee { // Parent

    public void empId() {
        System.out.println("Employee Id is: 1234");
    }

    public void empName() {
        System.out.println("Employee Name is: Bala");
    }

}
```

```java
public class Company extends Employee { // Child

    public void compId() {
        System.out.println("Company ID is: 2345");
    }

    public void compName() {
        System.out.println("Company Name is: CTS");
    }

    public static void main(String[] args) {
        Company c = new Company();
        c.compId();
        c.compName();
        c.empId();
        c.empName();
    }
}
```

2. Write a Code for Multilevel Inheritance

```java
public class Employee { // GrandParent

    public void empId() {
        System.out.println("Employee Id is: 1234");
    }

    public void empName() {
        System.out.println("Employee Name is: Bala");
    }
}
```

*Company.java

```java
package org.cts;

public class Company extends Employee { // Parent

    public void compId() {
        System.out.println("Company ID is: 2345");
    }

    public void compName() {
        System.out.println("Company Name is: CTS");
    }
}
```

```java
public class Client extends Company { // Child

    public void clientId() {
        System.out.println("Client Id is: 9876");
    }

    public static void main(String[] args) {
        Client c = new Client();
        c.clientId();
        c.compId();
        c.compName();
        c.empId();
        c.empName();
    }

}
```

3. Write a Code for Method Overloading

```java
public class Employee {

    public void empDetails(String empName) {
        System.out.println(empName);
    }

    public void empDetails(int empId, String empName, float empSalary) {
        System.out.println(empId + empName + empSalary);
    }

    public void empDetails(char empGender) {
        System.out.println(empGender);
    }

    public void empDetails(int pinCode, String empCity) {
        System.out.println(pinCode + empCity);
    }

    public void empDetails(String empCity, int pinCode) {
        System.out.println(empCity + pinCode);
    }
    public static void main(String[] args) {
        Employee e = new Employee();
        e.empDetails("bala");
        e.empDetails(123, "bala", 46528.12f);
        e.empDetails('M');
        e.empDetails(600028, "chennai");
        e.empDetails("chennai", 600028);
    }
}
```

4. Write a Code for Method Overriding

```java
public class RBIBank {

    public void fixed() {
        System.out.println("5%");
    }

    public void savings() {
        System.out.println("7%");
    }

}
```

```java
public class ICICIBank extends RBIBank {

    @Override
    public void savings() {

        System.out.println("8%");
    }

    public void deposite() {

        System.out.println("12%");

    }

    public static void main(String[] args) {

        ICICIBank bank = new ICICIBank();
        bank.fixed();
        bank.savings();
        bank.deposite();

    }

}
```

5. Write a Code for Interface

```java
public interface AxisBank {

    void savings();

    void deposit();

}
```

```java
public class ICICIBank implements AxisBank {

    @Override
    public void savings() {

        System.out.println("6%");

    }

    @Override
    public void deposit() {

        System.out.println("7%");
    }

    public static void main(String[] args) {

        ICICIBank bank = new ICICIBank();
        bank.savings();
        bank.deposit();

    }

}
```

6.Write a Code for Abstract Class

```java
public abstract class RBIBank {

    public abstract void fixed();

    public void savings() {
        System.out.println("7%");
    }

}
```

```java
public class ICICIBank extends RBIBank {

    @Override
    public void fixed() {

    }

    public static void main(String[] args) {

        ICICIBank bank = new ICICIBank();
        bank.fixed();
        bank.savings();

    }

}
```

7. Write a Code to archive Multiple Inheritance through Interface

```java
public interface Parent1 {

    void test1();

    void test2();

}
```

Parent2.java

```java
package org.test;

public interface Parent2 {

    void test2();

    void test3();

}
```

```java
public class ICICIBank implements Parent1, Parent2 {

    @Override
    public void test3() {

        System.out.println("Test3");
    }

    @Override
    public void test1() {

        System.out.println("Test1");
    }

    @Override
    public void test2() {

        System.out.println("Test2");
    }

    public static void main(String[] args) {

        ICICIBank bank = new ICICIBank();
        bank.test1();
        bank.test2();
        bank.test3();
    }

}
```

8. Create List insert Values and Iterate

```
import java.util.List;

public class Sample {
    public static void main(String[] args) {
        List<Integer> li = new ArrayList<Integer>();
        li.add(10);
        li.add(20);
        li.add(30);
        li.add(40);
        li.add(10);
        System.out.println(li);
        // Iterate Using Normal For Loop
        for (int i = 0; i < li.size(); i++) {
            Integer x = li.get(i);
            System.out.println(x);
        }
        // Iterate Using Enhanced For Loop
        for (Integer x : li) {
            System.out.println(x);
        }

    }
```

9. Create Set insert Values and Iterate

```
import java.util.Set;

public class Sample {
    public static void main(String[] args) {
        Set<Integer> li = new LinkedHashSet<Integer>();
        li.add(10);
        li.add(20);
        li.add(30);
        li.add(40);
        li.add(10);
        System.out.println(li);

        // Iterate Using Enhanced For Loop
        for (Integer x : li) {
            System.out.println(x);
        }

    }

}
```

10. Create Map insert Values and Iterate only the Entries

```java

import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class Sample {

    public static void main(String[] args) {

        Map<Integer, String> m = new LinkedHashMap<Integer, String>();

        m.put(10, "Java");
        m.put(20, "SQL");
        m.put(30, "Selenium");
        m.put(40, "SQL");
        m.put(50, "Ruby");
        m.put(10, "Python");

        System.out.println(m);

        // Iterate the Entries
        Set<Entry<Integer, String>> entrySet = m.entrySet();

        for (Entry<Integer, String> x : entrySet) {
            System.out.println(x);
        }
    }
}

```

11. Create Map insert Values and Iterate the Key and Values

```java

import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class Sample {

    public static void main(String[] args) {

        Map<Integer, String> m = new LinkedHashMap<Integer, String>();

        m.put(10, "Java");
        m.put(20, "SQL");
        m.put(30, "Selenium");
        m.put(40, "SQL");
        m.put(50, "Ruby");
        m.put(10, "Python");

        System.out.println(m);

        // Iterate the Keys & Values
        Set<Entry<Integer, String>> entrySet = m.entrySet();

        for (Entry<Integer, String> x : entrySet) {
            System.out.println(x.getKey());
            System.out.println(x.getValue());
        }
    }
}
```

12. Create Map insert Values and Iterate only the Keys

```java
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class Sample {

    public static void main(String[] args) {

        Map<Integer, String> m = new LinkedHashMap<Integer, String>();

        m.put(10, "Java");
        m.put(20, "SQL");
        m.put(30, "Selenium");
        m.put(40, "SQL");
        m.put(50, "Ruby");
        m.put(10, "Python");

        System.out.println(m);

        // Iterate the Keys only
        Set<Entry<Integer, String>> entrySet = m.entrySet();

        for (Entry<Integer, String> x : entrySet) {
            System.out.println(x.getKey());
        }
    }
}
```

13. Create Map insert Values and Iterate only the Values

```
2
3  import java.util.LinkedHashMap;
4  import java.util.Map;
5  import java.util.Map.Entry;
6  import java.util.Set;
7
8  public class Sample {
9
10     public static void main(String[] args) {
11
12         Map<Integer, String> m = new LinkedHashMap<Integer, String>();
13
14         m.put(10, "Java");
15         m.put(20, "SQL");
16         m.put(30, "Selenium");
17         m.put(40, "SQL");
18         m.put(50, "Ruby");
19         m.put(10, "Python");
20
21         System.out.println(m);
22
23         // Iterate the  Values only
24         Set<Entry<Integer, String>> entrySet = m.entrySet();
25
26         for (Entry<Integer, String> x : entrySet) {
27             System.out.println(x.getValue());
28         }
29     }
30 }
31
```

14. Write a code to handle the Exception using Try - Catch – Finally

```
2
3  public class Sample {
4
5      public static void main(String[] args) {
6
7          System.out.println(1);
8
9          System.out.println(2);
10
11         try {
12
13             System.out.println(3/0);
14
15         } catch (ArithmeticException e) {
16
17             System.out.println("Dont divide by Zero..");
18
19         }finally {
20
21             System.out.println("Success..");
22
23         }
24
25
26
27     }
28 }
29
```

15. Write any 5 Checked Exception and 5 Unchecked Exception Names

Checked Exception

- **FileNotFoundException**
- **IOException**
- **SQLException**
- **ClassNotFoundException**
- **InterruptedException**

Unchecked Exception

- **NullPointerException**
- **ArrayIndexOutOfBoundsException**
- **ClassCastException**
- **ArithmeticException**
- **StringIndexOutOfBoundsException**

16. Create Constructor and Pass the Argument

```
2
3  public class Employee {
4
5      public Employee() {
6
7          System.out.println("Default Constrcutor");
8      }
9
10     public Employee(String name) {
11
12         System.out.println("Para Constrcutor: "+name);
13     }
14
15     public Employee(int id, char gender) {
16
17         System.out.println("Para Constrcutor: "+id+gender);
18     }
19
20     public static void main(String[] args) {
21
22         Employee e = new Employee();
23         Employee e1 = new Employee("Bala");
24         Employee e2 = new Employee(1234, 'M');
25     }
26
27
28 }
29
```

17. Create Constructor and Pass the Argument using this()

```
3  public class Employee {
4
5      public Employee() {
6          this(1234, 'M');
7          System.out.println("Default Constrcutor");
8      }
9
10     public Employee(String name) {
11
12         System.out.println("Para Constrcutor: " + name);
13     }
14
15     public Employee(int id, char gender) {
16         this("Bala");
17         System.out.println("Para Constrcutor: " + id + gender);
18     }
19
20     public static void main(String[] args) {
21
22         Employee e = new Employee();
23
24     }
25
26 }
27
```

18. Create Constructor and Pass the Argument using super()

```java
public class Company {

    public Company() {
        System.out.println("Parent Default Constrcutor");
    }

    public Company(int id) {
        System.out.println("Para Default Constrcutor");
    }

}
```

```java
public class Employee extends Company{

    public Employee() {
        super(1234);
        System.out.println("Default Constrcutor");
    }

    public static void main(String[] args) {

        Employee e = new Employee();

    }

}
```

19. Write a code for Singleton Class

```java
public class Employee {

    static Employee e;

    private Employee() {
    }

    public static Employee getObject() {
        if (e == null) {
            e = new Employee();
        }
        return e;
    }

    public static void main(String[] args) {

        Employee e = getObject();
        System.out.println(System.identityHashCode(e));

        Employee e1 = getObject();
        System.out.println(System.identityHashCode(e1));

    }

}
```

```java
public class Sample {

    public static void main(String[] args) {

        Employee e = Employee.getObject();
        System.out.println(System.identityHashCode(e));

        Employee e1 = Employee.getObject();
        System.out.println(System.identityHashCode(e1));

    }

}
```

20. Create User defined List, Insert Values and Iterate

```java
public class Employee {

    private int empId;
    private String empName;

    public int getEmpId() {
        return empId;
    }

    public void setEmpId(int empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

}
```

```java
import java.util.ArrayList;

public class Sample {

    public static void main(String[] args) {

        List<Employee> li = new ArrayList<Employee>();

        Employee e1 = new Employee();
        e1.setEmpId(10);
        e1.setEmpName("Bala");

        Employee e2 = new Employee();
        e2.setEmpId(20);
        e2.setEmpName("Arun");

        li.add(e1);
        li.add(e2);

        for (int i = 0; i < li.size(); i++) {

            System.out.println(li.get(i).getEmpId());
            System.out.println(li.get(i).getEmpName());

        }

    }
```

21. Create User defined Set, Insert Values and Iterate

```java
public class Employee {

    private int empId;
    private String empName;

    public int getEmpId() {
        return empId;
    }

    public void setEmpId(int empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

}
```

```java

public class Sample {

    public static void main(String[] args) {

        Set<Employee> s = new LinkedHashSet<Employee>();

        Employee e1 = new Employee();
        e1.setEmpId(10);
        e1.setEmpName("Bala");

        Employee e2 = new Employee();
        e2.setEmpId(20);
        e2.setEmpName("Arun");

        s.add(e1);
        s.add(e2);

        for (Employee x : s) {
            System.out.println(x.getEmpId());
            System.out.println(x.getEmpName());
        }

    }

}
```

22. Create User defined Map, Insert Values and Iterate

```java
public class Employee {

    private int empId;
    private String empName;

    public int getEmpId() {
        return empId;
    }

    public void setEmpId(int empId) {
        this.empId = empId;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

}
```

```java
public class Sample {

    public static void main(String[] args) {

        Map<Integer, Employee> mp = new LinkedHashMap<Integer, Employee>();

        Employee e1 = new Employee();
        e1.setEmpId(10);
        e1.setEmpName("Bala");

        Employee e2 = new Employee();
        e2.setEmpId(20);
        e2.setEmpName("Arun");

        mp.put(1, e1);
        mp.put(2, e2);

        Set<Entry<Integer, Employee>> en = mp.entrySet();
        for (Entry<Integer, Employee> x : en) {

            System.out.println(x.getKey() + "===>Employee info");
            System.out.println(x.getValue().getEmpId());
            System.out.println(x.getValue().getEmpName());

        }

    }
```

23. Write a code for user Defined Exception

```java
public class EmployeeNotFoundException extends Exception{

    @Override
    public String getMessage() {

        String mess = " Employee Not Found In Unit1";
        return mess;
    }

}
```

```java
public class Employee extends Company{

    private void search() throws EmployeeNotFoundException {

        throw new EmployeeNotFoundException();

    }

    public static void main(String[] args) throws EmployeeNotFoundException {

        Employee e = new Employee();
        e.search();
    }

}
```

24. Write a code to read the Values from the File

```java

import java.io.File;
import java.io.IOException;
import java.util.List;

import org.apache.commons.io.FileUtils;

public class Sample1 {

    public static void main(String[] args) throws IOException {

        File file = new File("path/to/your/file.txt");

        List<String> lines = FileUtils.readLines(file);

        for (String x : lines) {

            System.out.println(x);

        }


    }

}

```

25. Write 10 Methods in String

length()
startsWith()
endsWith()
toUpperCase()
toLowerCase()
indexOf()
lastIndexOf()
charAt(index)
isEmpty()
contains()
equals()
equalsIgnoreCase()
replace()
replaceAll()
subString(start Index)
subString(start Index,end Index)
concat()
trim()

26. Write a code to insert 5 values as number in Array and Iterate by using Enhanced for loop

```
public class Employee {

    public static void main(String[] args) {

        int a[] = new int[5];

        a[0] = 10;
        a[1] = 20;
        a[2] = 30;
        a[3] = 40;
        a[4] = 50;

        for (int i : a) {

            System.out.println(i);

        }

    }

}
```

27 Write a code to declare a String in 4 ways

```java
public class Sample1 {

    public static void main(String[] args)  {

    String s = "Hello";

    String s1 = new String("Hello");

    StringBuffer s2 = new StringBuffer("Hello");

    StringBuilder s3 = new StringBuilder("Hello");


    }

}

```

28.Write a code to change given Array to Array List

```java

import java.util.ArrayList;

public class Sample1 {

    public static void main(String[] args) {

        int a[] = new int[5];

        a[0] = 10;
        a[1] = 20;
        a[2] = 30;
        a[3] = 40;
        a[4] = 50;

        ArrayList<Integer> al = new ArrayList<Integer>();

        for (int i = 0; i < 5; i++) {
            al.add(a[i]);

        }
        System.out.println(al);
    }

}

```

29. Write a code to remove the unused objects using finalize method

```
public class Employee {

    public Employee() {
        System.out.println("object created");
    }

    @Override
    protected void finalize() throws Throwable {
        System.out.println("object destroyed");
    }

    public static void main(String[] args) {

        Employee e = new Employee();
        e = null;

        System.gc();

    }
}
```