1. Define Unicode?

Unicode system is a global standard that is used for character encoding of 16-bit characters.

Example :- Unicode Character “A” (U+0041)

1. What is object class?

**Class is used as a template for declaring and creating the objects. An object is an instance of a class.**

1. Define Super keyword?

**Super is a reference variable that is used to refer to the present class objet.**

**It comes under the concept of inheritance**

**class A{**

**int a1 = 33;}**

**class B extends A{**

**int a1 = 44;**

**System.out.print(super.a);**

**}**

1. Define this keyword/ static member or keyword

**‘this’ is a reference variable that refers to the current object.**

**//Java code for using 'this' keyword to**

**//refer current class instance variables**

**class Test**

**{**

**int a;**

**int b;**

**// Parameterized constructor**

**Test(int a, int b)**

**{**

**this.a = a;**

**this.b = b;**

**}**

**void display()**

**{**

**//Displaying value of variables a and b**

**System.out.println("a = " + a + " b = " + b);**

**}**

**public static void main(String[] args)**

**{**

**Test object = new Test(10, 20);**

**object.display();**

**}**

**}**

1. What is object code
2. Differentiate between byte code and object code

**Bytecode is an artificial machine code for a virtual machine (VM), such as the Java Virtual Machine (JVM), whereas object code is a code produced by a compiler or assembler**.

1. What is JRE/JVM?

**JRE stands for “Java Runtime Environment” and may also be written as “Java RTE.” The Java Runtime Environment provides the minimum requirements for executing a Java application; it consists of the Java Virtual Machine (JVM), core classes, and supporting files.**

**3.**[**JVM (Java Virtual Machine)**](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/)**is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line, hence it is also known as an**[**i*nterpreter***](https://www.geeksforgeeks.org/compiler-vs-interpreter-2/)**.**

1. Runtime Polymorphism and compile time polymorphism

| Compile Time Polymorphism | Run time Polymorphism |
| --- | --- |
| In Compile time Polymorphism, the call is resolved by the compiler. | In Run time Polymorphism, the call is not resolved by the compiler. |
|  |  |
| Method overloading is the compile-time polymorphism where more than one methods share the same name with different parameters or signature and different return type. | Method overriding is the runtime polymorphism having the same method with same parameters or signature but associated withcompared, different classes. |
| It is achieved by function overloading and operator overloading. | It is achieved by virtual functions and pointers. |
| It provides fast execution because the method that needs to be executed is known early at the compile time. | It provides slow execution as compare to early binding because the method that needs to be executed is known at the runtime. |
| Compile time polymorphism is less flexible as all things execute at compile time. | Run time polymorphism is more flexible as all things execute at run time. |
| Inheritance is not involved. | Inheritance is involved. |

1. Overloading and Overriding
2. he differences between Method Overloading and Method Overriding in Java are as follows:

| Method Overloading | Method Overriding |
| --- | --- |
| Method overloading is a compile-time polymorphism. | Method overriding is a run-time polymorphism. |
| It helps to increase the readability of the program. | It is used to grant the specific implementation of the method which is already provided by its parent class or superclass. |
| It occurs within the class. | It is performed in two classes with inheritance relationships. |
| Method overloading may or may not require inheritance. | Method overriding always needs inheritance. |
| In method overloading, methods must have the same name and different signatures. | In method overriding, methods must have the same name and same signature. |
| In method overloading, the return type can or can not be the same, but we just have to change the parameter. | In method overriding, the return type must be the same or co-variant. |
| Static binding is being used for overloaded methods. | Dynamic binding is being used for overriding methods. |
| Poor Performance due to compile time polymorphism. | It gives better performance. The reason behind this is that the binding of overridden methods is being done at runtime. |
| Private and final methods can be overloaded. | Private and final methods can’t be overridden. |
| Argument list should be different while doing method overloading. | Argument list should be same in method overriding. |

1. Features of oops
2. Classes
3. Objects
4. Data Abstraction
5. Encapsulation
6. Inheritance
7. Polymorphism