1. **How does compiler differentiate overloaded methods from duplicate methods?**

**Answer:** The Java compiler differentiates overloaded methods from duplicate methods based on their method signatures. The method signature includes the method name and the parameter types. Overloaded methods have the same name but different types or arrangement of parameter, allowing the compiler to distinguish between them. Duplicate methods, on the other hand, have the same name and the same parameter types, which leads to a compilation error.

1. **Write a java method that checks whether all the characters in a given string are vowels or not.**

**Answer:**

import java.util.Scanner;

public class vowelsChecker {

public static void main(String[] args) {

Scanner scn = new Scanner (System.***in***);

System.***out***.println("Input your String: ");

String word = scn.nextLine();

boolean vowels = false;

for (int i = 0; i < word.length(); i++ ) {

char c = word.charAt(i);

if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {

vowels = true;

break;

}

else {

vowels = false;

}

}

if (vowels) {

System.***out***.println("There is vowels in the String");

}

else {

System.***out***.println("There is no vowel in the String");

}

scn.close();

}

}

1. **Describe dynamic and static polymorphism.**

**Answer:**

Dynamic polymorphism, or runtime polymorphism, refers to the ability to invoke different method implementations based on the actual type of an object. It is achieved through method overriding and allows for flexibility in code design by treating objects of different classes as objects of a common superclass or interface. The specific method to be executed is determined at runtime using virtual method invocation.

Static polymorphism, or compile-time polymorphism, is achieved through method overloading. It allows multiple methods with the same name but different parameter lists to coexist within a class. The appropriate method implementation is determined at compile-time based on the method signature and the arguments passed to the method. Static polymorphism promotes code reuse and clarity by providing different method implementations for different parameter combinations.