OOP ASSIGNMENT 3

B.AJAY KUMAR

1602-22-737-004

IT A

2. The 'HashSet' in Java's Collection framework utilizes a hash table structure, employing a 'HashMap' as its underlying storage. When adding elements, it computes their hash codes to determine their storage location within the 'HashMap.' If collisions occur (where multiple elements share the same storage location), a chaining mechanism, like linked lists or trees, is employed to manage them efficiently. 'HashSet' guarantees element uniqueness by verifying hash codes first and then using the 'equals()' method for further confirmation within the storage buckets. This structure ensures that operations like retrieval, deletion, and searching within 'HashSet' are generally swift (nearly constant time), making it well-suited for scenarios where quick access to unique elements is essential.

Program

import java.util.HashSet;

import java.util.Iterator;

public class HashSetIteratorExample {

public static void main(String[] args) {

// Creating a HashSet

HashSet<String> names = new HashSet<>();

// Adding elements to the HashSet

names.add("Alice");

names.add("Bob");

names.add("Charlie");

names.add("David");

// Creating an Iterator to traverse the HashSet

Iterator<String> iterator = names.iterator();

// Using Iterator to access elements

System.out.println("Elements in the HashSet:");

while (iterator.hasNext()) {

String name = iterator.next();

System.out.println(name);

}

}

}

3. (a)Limitations of AWT:

1.Platform Dependency: AWT components are rendered using native platform elements, making their appearance and behavior platform-dependent. This can cause inconsistency across different operating systems.

2.Limited Components: AWT has a limited set of components compared to Swing. It lacks more advanced components like tables, trees, and customizable components, limiting GUI design options.

3.Less Customization: AWT components have minimal customization options in terms of look and feel. This restricts developers from extensively modifying the appearance of components.

4.Performance: Due to the reliance on native components, AWT's performance can suffer as it relies heavily on the host system's resources.

AWT and Swing Applications:

1.Component Set: AWT provides a basic set of components (buttons, text fields), while Swing offers an extended set (tables, trees, sliders), allowing more complex and customizable GUIs.

2.Look and Feel: Swing provides a consistent look and feel across different platforms as it's rendered purely in Java, unlike AWT which relies on native components.

3.Customization: Swing components offer greater customization options in terms of appearance and behavior compared to AWT, enabling developers to create more sophisticated user interfaces.

4.Performance: Swing might have better performance compared to AWT in certain cases due to its lightweight nature and the ability to bypass native components.

5.Portability: Swing applications tend to be more portable because they are not as reliant on the underlying native system as AWT, which makes Swing applications consistent across different platforms.

(b)

import java.applet.Applet;

import java.awt.Graphics;

public class ParameterApplet extends Applet {

String message;

public void init() {

// Retrieving the parameter named "text" from the HTML tag

message = getParameter("text");

if (message == null) {

message = "No message provided";

}

}

public void paint(Graphics g) {

g.drawString(message, 20, 20);

}

}

<!DOCTYPE html>

<html>

<head>

<title>Applet Parameter Example</title>

</head>

<body>

<applet code="ParameterApplet.class" width="300" height="100">

<!-- Passing a parameter named "text" with a value -->

<param name="text" value="Hello, Applet! This is a parameter.">

Your browser does not support Java.

</applet>

</body>

</html>

OUTPUTS OF QUESTION 1







