Java

Scopes & GUI

Tobias Hanf, Manik Khurana

24. Januar 2022

Java-Course

Overview

Recursion in Java

Recursion

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples - Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc.

Recursion Contd.

Approach 1:

```
        approach(1) - Simply adding one by one
        f(n) = 1 + 2 + 3 + .... + n

```

Approach 2:

```
        approach(2) - Mathematical expression
        f(n) = 1                n=1
        f(n) = n + f(n-1)       n>1

```

Example

In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

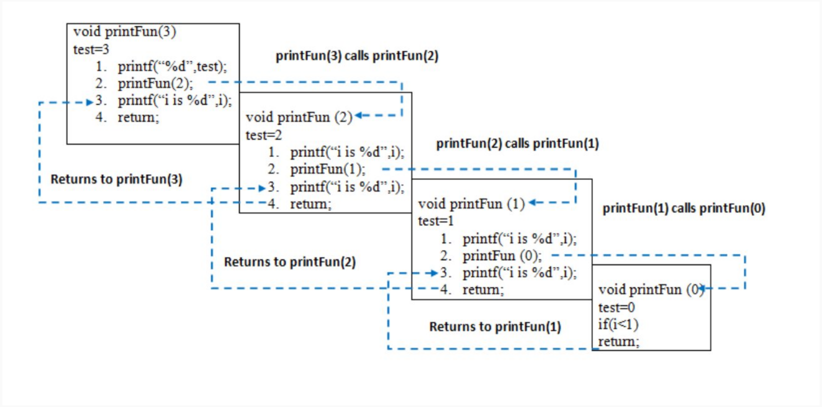
```
int fact(int n)
{
if (n < = 1) // base case
return 1;
else
return n*fact(n-1);
}

```

In the above example, the base case is defined and larger value of number can be solved by converting to smaller one till base case is reached.

Recursion Example

```java
// A Java program to demonstrate working of recursion
class GFG {
    static void printFun(int test)
    {
        if (test < 1)
        return;
        else {
            System.out.printf("%d ", test);
            printFun(test - 1); // statement 2
            System.out.printf("%d ", test);
            return;
        }}
    // Driver Code
    public static void main(String[] args){
        int test = 3;
        printFun(test);
    }
}
```

Flow of program



Why Stack Overflow error occurs in recursion?

Happens if the base case is not reached or not defined.

```
        int fact(int n)
        {
            // wrong base case (it may cause stack overflow)
    .
            if (n == 100)
            return 1;
            else
            return n*fact(n-1);
        }

```

If fact(10) is called, it will call fact(9), fact(8), fact(7) and so on but the number will never reach 100. So, the base case is not reached. If the memory is exhausted by these functions on the stack, it will cause a stack overflow error.

Programming exercise

Refer to fib.java code

Scopes

Visiabilities

```
class MyGreatClass {

    //Attributes are public by default
    Car myCar;

    //Public are available in every part of our code.
    public Cat myCat;

    //Private Attributes can only be acced via a method
    private House myHouse;
}


```

For

```java
    for(int i = 0; i <= 100; i++) {
        int b = 3;
        System.out.println(i);
        System.out.println(b);
    }

```

b will be redefined in every round of the loop and is only available in the for loop.

The scope is created at the begining and destroyed at the end of each round.

Examples

```java
    public class myClass {
        private int a;

        public myClass(int a) {
            this.a = a;
        }
    }

```

Use nearest definition.

In one scope every variable name can be defined only one time.

GUI

Window

Windows are created by creating a JFrame object.

```
        // Create a new window
        JFrame window = new JFrame();

        // Set its title and size
        window.setTitle("Guestbook");
        window.setSize(500, 500);

    // Show the window
    window.setVisible(true);

```

Window

You can add panels and other elements to the window.

```java
        // Add a blue background panel
        JPanel backgroundPanel = new JPanel();
        backgroundPanel.setBackground(Color.BLUE);
        window.add(backgroundPanel);

```

More advanced panels are available (JSplitPane, JScrollPane, JTabbedPane etc..)

Menus

Menus are used to display multiple possible actions to the user.

```
        // Menu Bar holds all menus
    MenuBar bar = new MenuBar();

    // Create new menu which holds menu items and submenus
        Menu menu = new Menu("File");

    // Create new menu item
        MenuItem item = new MenuItem("Create new User");

    // Combine everything
        menu.add(item);
        bar.add(menu);
        window.setMenuBar(bar);

```

Actions

In order to respond to button presses or other changes in the UI, you need to add listeners

```
    // Create menu item
        MenuItem item = new MenuItem("Create new User");

    // Add listener
        item.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                userManager.addUser();
            }
        });

```

Actions

Another example for a list

```
    // Create list with single selection option.
        this.userList = new JList();
        this.userList.setSelectionMode(ListSelectionModel.
    SINGLE_SELECTION);

    // Add listener
        this.userList.addListSelectionListener(new
    ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                int selectedIndex = this.userList.
    getSelectedIndex();
            }
        });

```