PSLG Week 13

Ran by Amy & Ben

DLSH

ICTLC Online QR Code:

GitHub

Today's Agenda:

- 2D Arrays
- Inheritance and Polymorphism
- Stacks & Queues
- HashMaps

2D Arrays syntax

Declaration:

```
int[][] name = new int[r][c];
```

Functions:

```
name[r][c] = 9;

Arrays.fill(name[r][c]);
```

Stacks & Queues Syntax

Stacks:

Initialisation

Stack<Type> name = new Stack<>();

Useful Methods

name.push(item);

name.pop();

LIFO

Queues:

Initialisation

Queue<Type> name = new LinkedList<>();

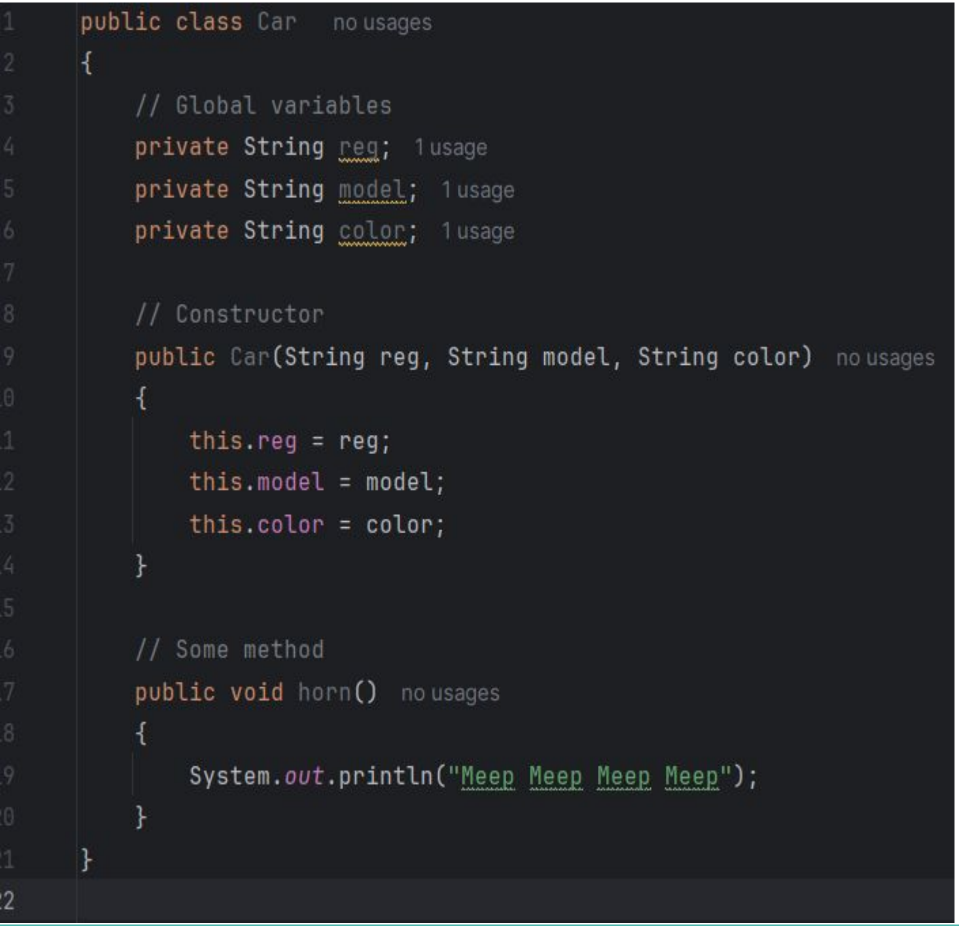
Useful Methods

name.offer(item);

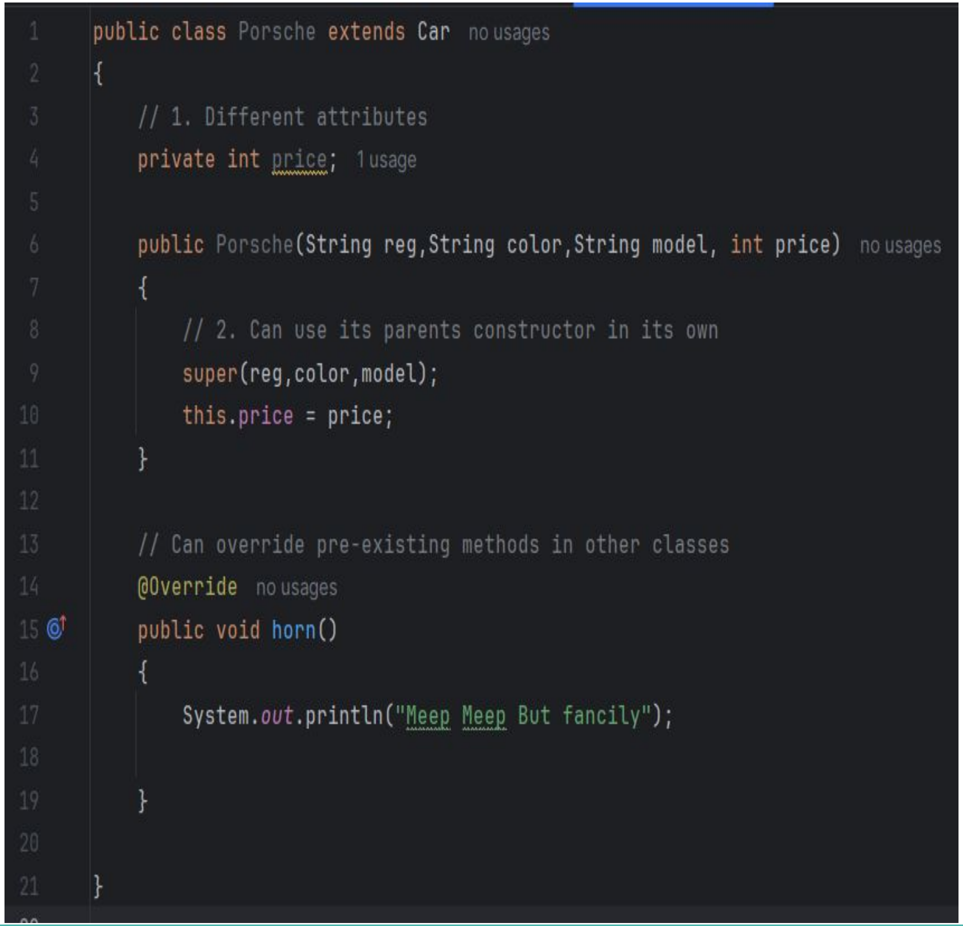
name.poll();

FIFO

Inheritance and Polymorphism Syntax



```
public class Car   no usages
{
    // Global variables
    private String reg;   1 usage
    private String model;   1 usage
    private String color;   1 usage

    // Constructor
    public Car(String reg, String model, String color)   no usages
    {
        this.reg = reg;
        this.model = model;
        this.color = color;
    }

    // Some method
    public void horn()   no usages
    {
        System.out.println("Meep Meep Meep Meep");
    }
}
```



```
public class Porsche extends Car   no usages
{
    // 1. Different attributes
    private int price;   1 usage

    public Porsche(String reg,String color,String model, int price)   no usages
    {
        // 2. Can use its parents constructor in its own
        super(reg,color,model);
        this.price = price;
    }

    // Can override pre-existing methods in other classes
    @Override   no usages
    public void horn()
    {
        System.out.println("Meep Meep But fancily");

    }

}
```

First question :D

Make a class called fill D-struct

Make subclasses called FillD-StructStack, FillD-StructQueue and FillD-StructArray

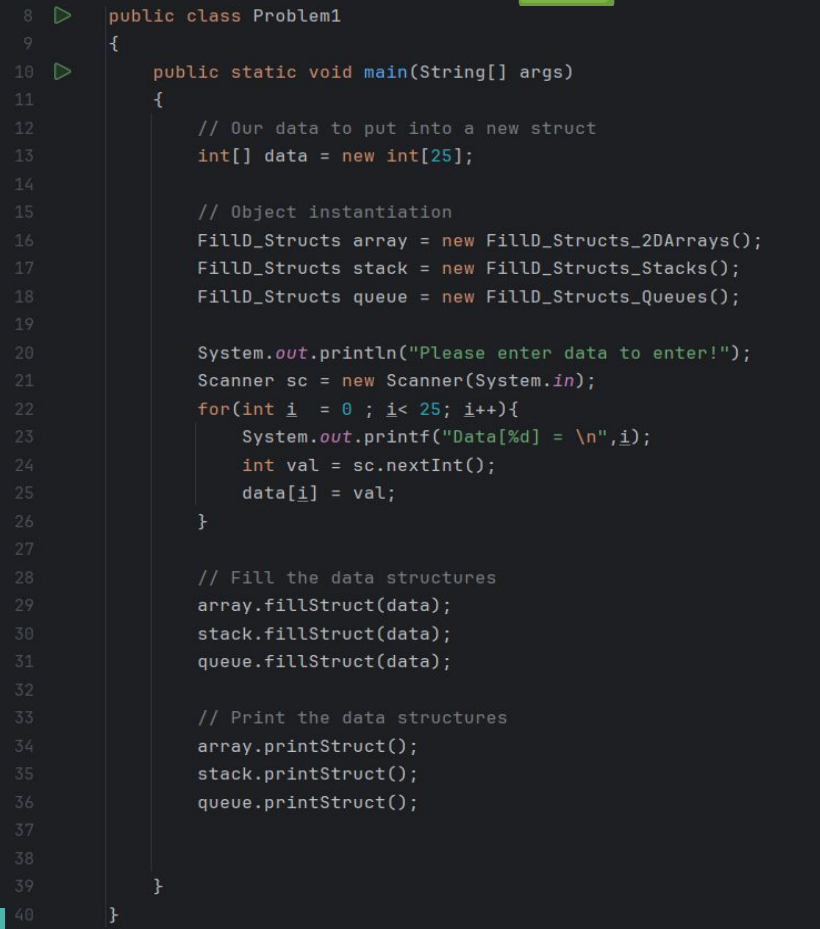
Make each class have the specified data-structure in the name as class attribute.

Each subclass will also have a overridden method from the fillD-Struct class that will fill the specified data structure with whatever data you enter via scanner

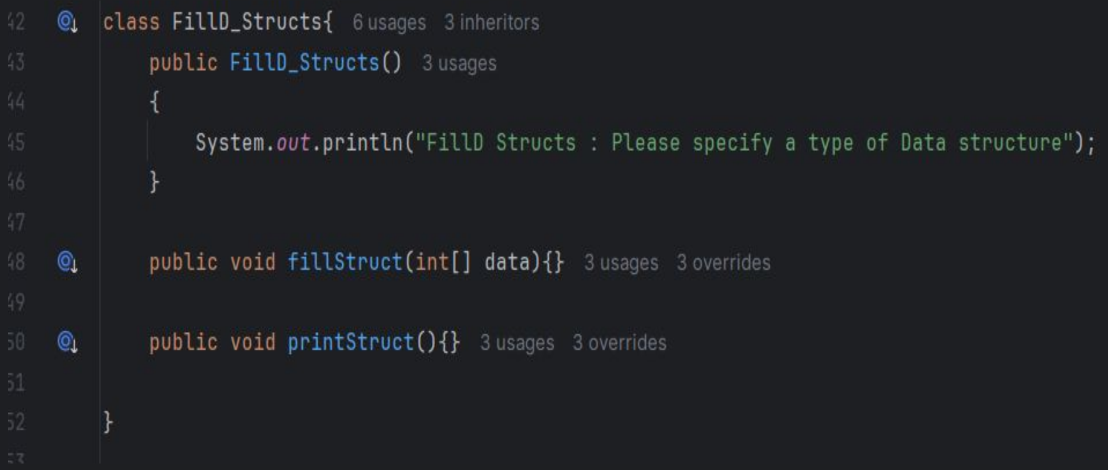
(The FillD-StructArray class will use a static 5x5 array).

Test this program by making a main driver and printing out the results of filling each class with the data.

First Solution ✅

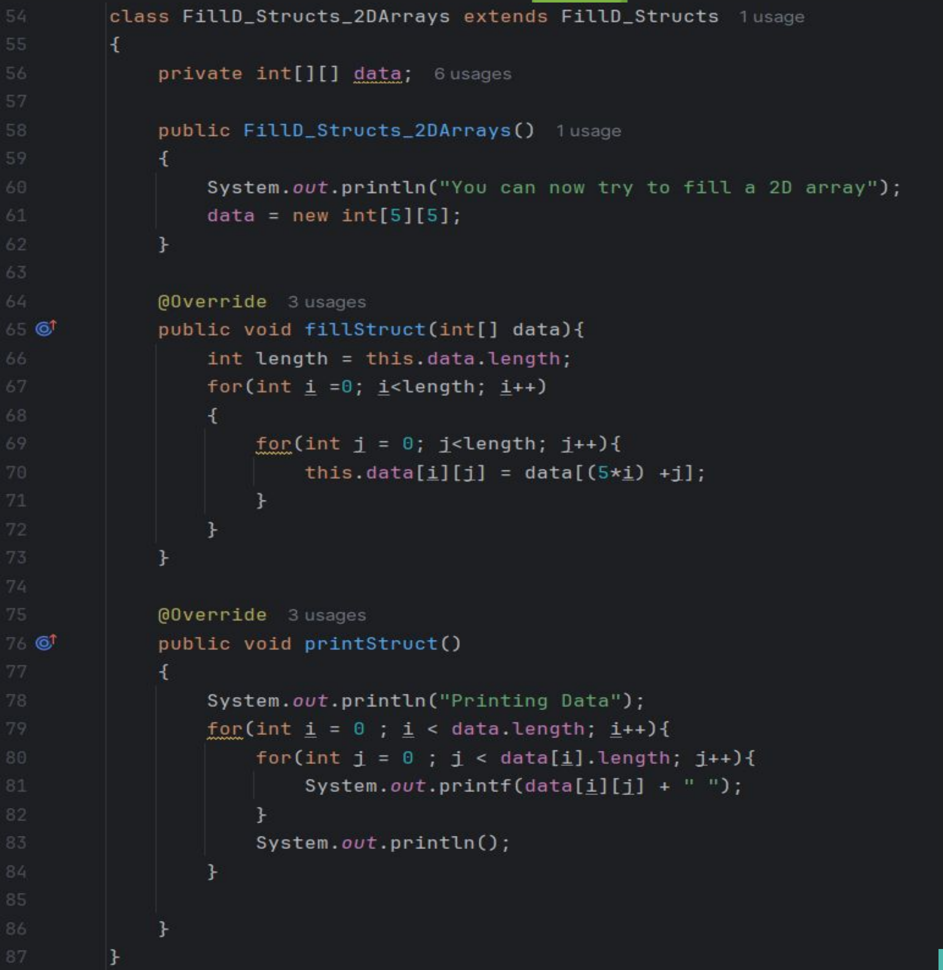


```java
public class Problem1
{
    public static void main(String[] args)
    {
        // Our data to put into a new struct
        int[] data = new int[25];

        // Object instantiation
        FillD_Structs array = new FillD_Structs_2DArrays();
        FillD_Structs stack = new FillD_Structs_Stacks();
        FillD_Structs queue = new FillD_Structs_Queues();

        System.out.println("Please enter data to enter!");
        Scanner sc = new Scanner(System.in);
        for(int i  = 0 ; i< 25; i++){
            System.out.printf("Data[%d] = \n",i);
            int val = sc.nextInt();
            data[i] = val;
        }

        // Fill the data structures
        array.fillStruct(data);
        stack.fillStruct(data);
        queue.fillStruct(data);

        // Print the data structures
        array.printStruct();
        stack.printStruct();
        queue.printStruct();


    }
}
```

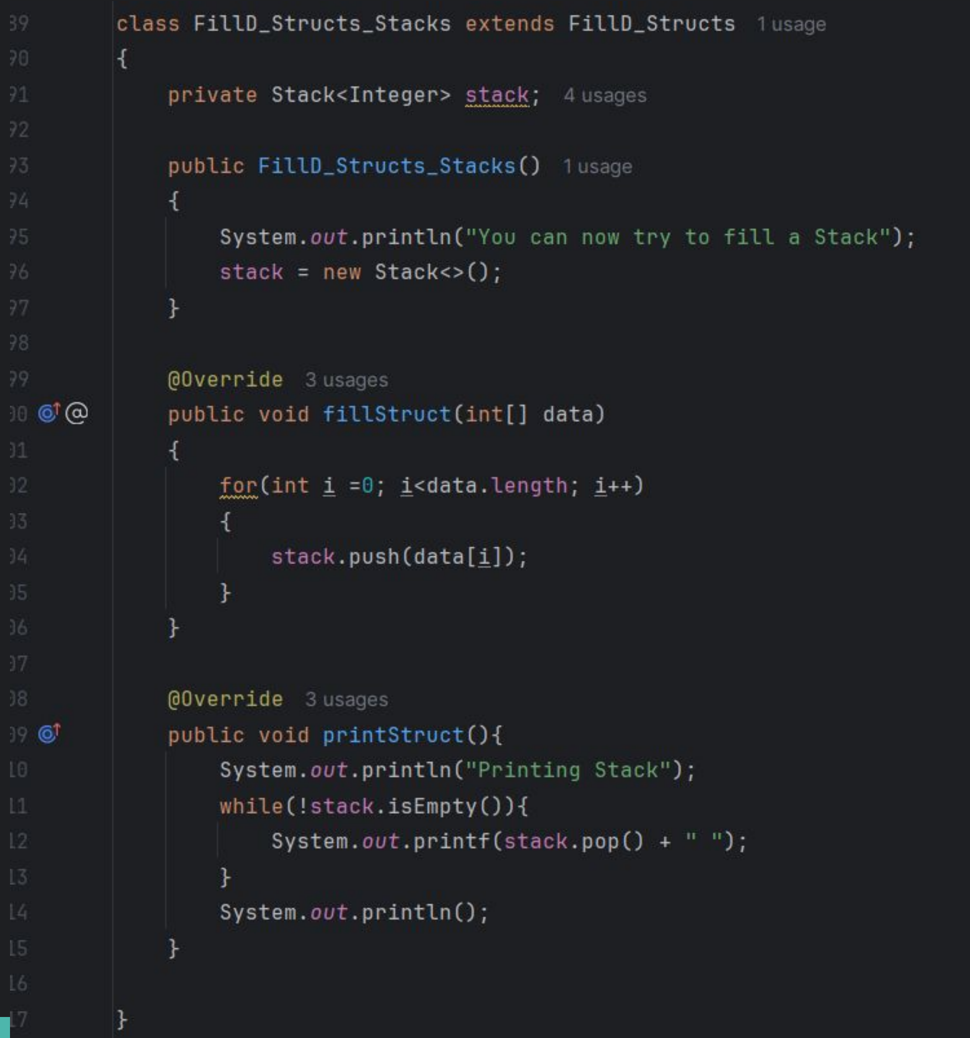


```java
class FillD_Structs{  6 usages  3 inheritors
    public FillD_Structs()  3 usages
    {
        System.out.println("FillD Structs : Please specify a type of Data structure");
    }

    public void fillStruct(int[] data){}  3 usages  3 overrides

    public void printStruct(){}  3 usages  3 overrides

}
```

First Solution ✅

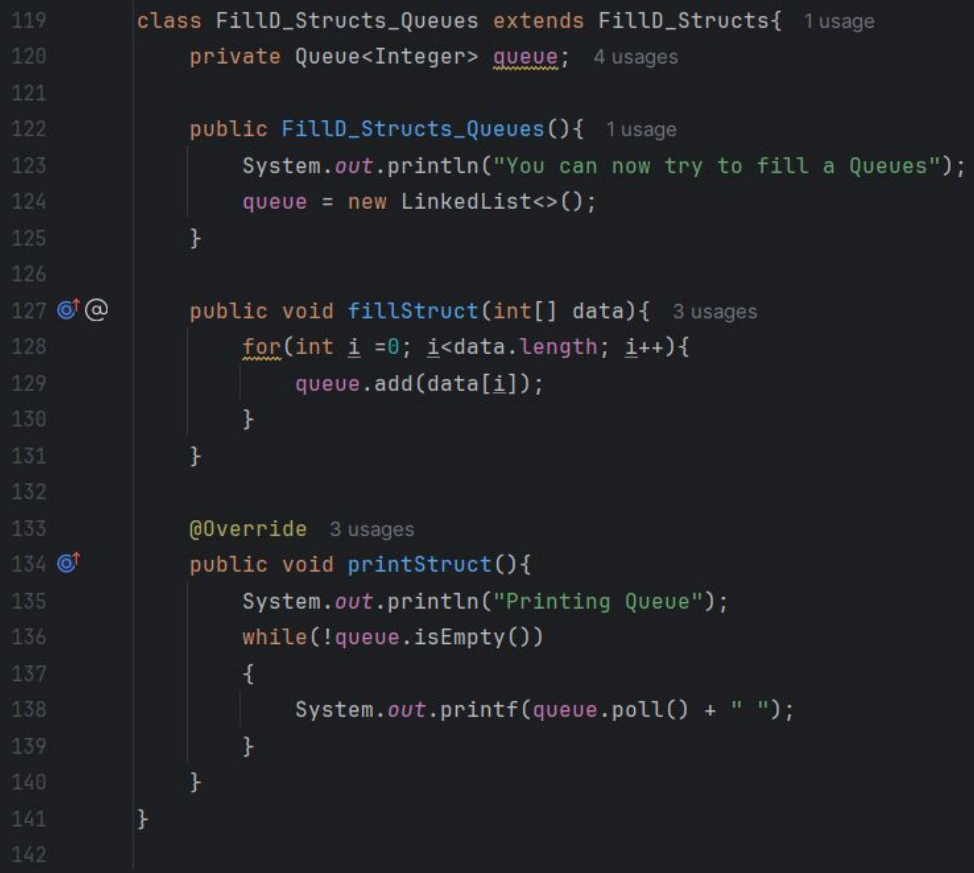


```java
class FillD_Structs_2DArrays extends FillD_Structs  1 usage
{
    private int[][] data;  6 usages

    public FillD_Structs_2DArrays()  1 usage
    {
        System.out.println("You can now try to fill a 2D array");
        data = new int[5][5];
    }

    @Override  3 usages
    public void fillStruct(int[] data){
        int length = this.data.length;
        for(int i =0; i<length; i++)
        {
            for(int j = 0; j<length; j++){
                this.data[i][j] = data[(5*i) +j];
            }
        }
    }

    @Override  3 usages
    public void printStruct()
    {
        System.out.println("Printing Data");
        for(int i = 0 ; i < data.length; i++){
            for(int j = 0 ; j < data[i].length; j++){
                System.out.printf(data[i][j] + " ");
            }
            System.out.println();
        }

    }
}
```



```java
class FillD_Structs_Stacks extends FillD_Structs  1 usage
{
    private Stack<Integer> stack;  4 usages

    public FillD_Structs_Stacks()  1 usage
    {
        System.out.println("You can now try to fill a Stack");
        stack = new Stack<>();
    }

    @Override  3 usages
    public void fillStruct(int[] data)
    {
        for(int i =0; i<data.length; i++)
        {
            stack.push(data[i]);
        }
    }

    @Override  3 usages
    public void printStruct(){
        System.out.println("Printing Stack");
        while(!stack.isEmpty()){
            System.out.printf(stack.pop() + " ");
        }
        System.out.println();
    }

}
```

First Solution ✅



```
class FillD_Structs_Queues extends FillD_Structs{  1 usage
    private Queue<Integer> queue;  4 usages

    public FillD_Structs_Queues(){  1 usage
        System.out.println("You can now try to fill a Queues");
        queue = new LinkedList<>();
    }

    public void fillStruct(int[] data){  3 usages
        for(int i =0; i<data.length; i++){
            queue.add(data[i]);
        }
    }

    @Override  3 usages
    public void printStruct(){
        System.out.println("Printing Queue");
        while(!queue.isEmpty())
        {
            System.out.printf(queue.poll() + " ");
        }
    }
}
```

HashMaps

- Maps are a collection of key-value pairs AKA a key maps to a value.

Syntax:

Map<KeyType, ValType>name = new HashMap<>();

Operations:

name.put(Key, Val); //adding elements

name.get(Key); //gets val at specified key

name.remove(Key); //removes val at specified key

name.replace(Key, NewVal); //replaces the val at specified key with new val

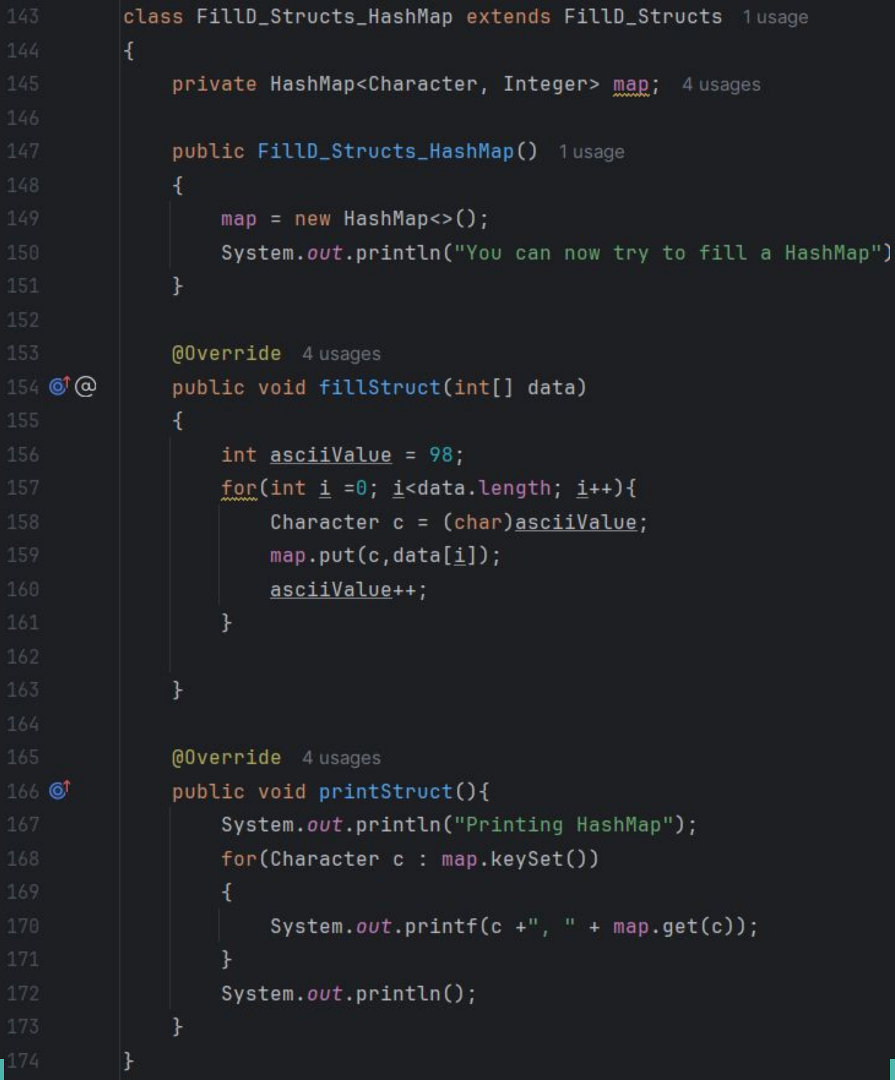
HashMap problem

Add a new FillD-Struct subclass called FillD-Struct_HashMap and ensure the following
Each key is a letter of the alphabet in uppercase, Once all uppercase letters have been expended each following key will be a number
(hint use characters and ascii values)

Each key will be paired to a piece of data as in the previous questions solution.

A print method will also be necessary which prints out each key and its corresponding value

Solution



```java
class FillD_Structs_HashMap extends FillD_Structs  1 usage
{
    private HashMap<Character, Integer> map;  4 usages

    public FillD_Structs_HashMap()  1 usage
    {
        map = new HashMap<>();
        System.out.println("You can now try to fill a HashMap")
    }

    @Override  4 usages
    public void fillStruct(int[] data)
    {
        int asciiValue = 98;
        for(int i =0; i<data.length; i++){
            Character c = (char)asciiValue;
            map.put(c,data[i]);
            asciiValue++;
        }

    }

    @Override  4 usages
    public void printStruct(){
        System.out.println("Printing HashMap");
        for(Character c : map.keySet())
        {
            System.out.printf(c +", " + map.get(c));
        }
        System.out.println();
    }
}
```