



```
// Fixed_STK.java package Assign6.Part1;
public class Fixed_STK implements
Interface_STK {
private int arr[];
private int top;
private int size;
@Override
public int pop(){
if(isEmpty()){
System.out.println("Stack is empty");
return -1;
} else{
int item = arr[top];          top--
;          return item;
}
} public void
push(int element){
if(isFull()){
System.out.println("Stack is full");
}
else{
top++;
arr[top] = element;
System.out.printf("%d pushed\n", element);
}
} public Fixed_STK(int[] arr,
int top){
this.arr = arr;
this.top = top;
} public void
display(){
if(isEmpty()){
System.out.println("Stack is empty");
}
else{
for(int i = top; i >= 0; i--){
System.out.println(arr[i]);
}
}
} public boolean
isEmpty(){
```

```
        return top == -1;
    }    public boolean isFull(){        return top
== size - 1;
    }
}
```

```
//Growable_stk.java package Assign6.Part1;
import
java.util.ArrayList;

public class Growable_stk implements Interface_STK{
ArrayList<Integer> stack;    int top;    public
Growable_stk(){        top = -1;        stack = new
ArrayList<>(5);
    }

    @Override    public boolean isEmpty(){
return top == -1;
    }

    @Override    public int pop(){
if(top == -1){
        System.out.println("Stack is empty");        return -1;
    }    else{        return
stack.remove(top--);
    }
    }

    @Override    public void push(int element){
stack.add(++top, element);
    }

    @Override
    public boolean isFull(){
        System.out.println("Stack is not growable");        return false;
    }
}
```

```

    }

    @Override    public
void display() {
if (isEmpty()) {
    System.out.println("Stack is empty");
} else {
    System.out.println("Stack elements: ");
for (int i = top; i >= 0; i--) {
    System.out.print(stack.get(i) + " ");
}
    System.out.println();
}
}
}

```

```

package Assign6.Part1;  public interface Interface_STK
{    public int pop();    public void push(int
element);    public void display();    public boolean
isEmpty();    public boolean isFull();
}

```

```

//Main.java package Assign6.Part1;
public class Main {    public static void
main(String[] args) {
    Growable_stk g = new Growable_stk();
g.push(1);
    g.push(2);
    g.push(3);
    g.push(4);
    g.push(5);
    g.display();
    g.push(6);
    g.display();    System.out.println("Popped
Element:" + g.pop());    g.display();

    System.out.println("Popped Element:" +
g.pop());    g.display();

    System.out.println("Popped Element:" + g.pop());
g.display();

```

```
        System.out.println("Popped Element:" + g.pop());
g.display();

        System.out.println("Popped Element:" + g.pop());
g.display();

        System.out.println("Popped Element:" + g.pop());
g.display();
    }
}
```

## Part 2

```
// DecoyDuck.java package
Assign6.Part2;

// DecoyDuck class extends Duck class to represent
a specific type of duck public class DecoyDuck
extends Duck {

    // Constructor for DecoyDuck initializes behaviors
public DecoyDuck() {
    // Set fly behavior to unable to
fly        flyBehaviour = new FlyNoWay();
// Set quack behavior to squeak
quackBehaviour = new Squeak();
    // Set swim behavior to unable to swim
swimBehaviour = new SwimNoWay();
    }

    // Method to display DecoyDuck
@Override
public void display() {
    System.out.println("I'm a decoy duck!!! lmao");
}
}
```

```
//Duck.java package Assign6.Part2;

// Duck class serves as the abstract base class
for different types of ducks abstract public class
Duck {
```

```
// Attributes to hold behaviors
FlyBehaviour flyBehaviour;
QuackBehaviour quackBehaviour;
SwimBehaviour swimBehaviour;

// Method to set fly behavior dynamically
public void setFlyBehaviour(FlyBehaviour fb) {
    flyBehaviour = fb;
}

// Method to set quack behavior dynamically
public void setQuackBehaviour(QuackBehaviour qb) {
    quackBehaviour = qb;
}

// Method to set swim behavior dynamically
public void setSwimBehaviour(SwimBehaviour sb) {
    swimBehaviour = sb;
}

// Abstract method for displaying duck
abstract public void display();

// Method to perform fly behavior
public void performFly() {
    flyBehaviour.fly();
}

// Method to perform quack behavior
public void performQuack() {
    quackBehaviour.quack();
}

// Method to perform swim behavior
public void performSwim() {
    swimBehaviour.swim();
}
}
```

```
// Quack.java package Assign6.Part2;

// Quack class implements the QuackBehaviour interface
to represent quacking behavior
public class Quack implements QuackBehaviour {

    // Method implementation for quacking behavior
    @Override public void quack() {
        System.out.println("Duck says Quack Quack
lmao"); // Print a message indicating the duck is
quacking    }

}
```

```
// Squeak.java package Assign6.Part2;

// Squeak class implements the QuackBehaviour interface
to represent squeaking behavior public class Squeak
implements QuackBehaviour {
    // Method implementation for quacking behavior
    @Override public void quack() {
        System.out.println("Duck only Squeaks!!"); //
Print a message indicating the duck is squeaking
    }
}
```

```
// RedHeadDuck.java package
Assign6.Part2;
// RedHeadDuck class extends Duck class to represent a
specific type of duck public class RedHeadDuck extends
Duck {

    // Constructor for RedHeadDuck initializes
behaviors public RedHeadDuck() {
        // Set fly behavior to fly with wings
flyBehaviour = new FlyWithWings(); //
Set quack behavior to quack
quackBehaviour = new Quack();
```

## Suyash Tambe Assignment 6

```
        // Set swim behavior to swim
swimBehaviour = new Swim();
    }

    // Method to display RedHeadDuck
    @Override
    public void display() {
        System.out.println("I'm a red head duck!!!
Lmao"); // Print a message indicating the duck is a red
head duck
    }
}
```

```
// Main.java
package Assign6.Part2;

// Main class to test duck behaviors public
class Main {    public static void
main(String[] args) {
    // Create a RedHeadDuck object
    RedHeadDuck redHeadDuck = new RedHeadDuck();
    // Display information about RedHeadDuck
redHeadDuck.display();

    // Perform swim, fly, and quack behaviors
of RedHeadDuck    redHeadDuck.performSwim();
redHeadDuck.performFly();
redHeadDuck.performQuack();

    // Create a DecoyDuck object
    DecoyDuck decoyDuck = new DecoyDuck();
    // Display information about DecoyDuck
decoyDuck.display();

    // Perform fly, quack, and swim behaviors of
DecoyDuck
    decoyDuck.performFly();
decoyDuck.performQuack();
decoyDuck.performSwim();
    }
}
```

Output:



## Suyash Tambe Assignment 6

```
User\workspaceStorage\339cf22f19ad90c4ebe0eb489dcf6574\redhat.java\jdt
I'm a red head duck!!! Lmao
I can quackily swim. lmao
Flying with wings!!
Duck says Quack Quack lmao
I'm a decoy duck!!! lmao
Cannot fly.....lol
Duck only Squeaks!!
I can't quackily swim. lmao
PS C:\Users\Suyash Tambe\Desktop\PIJ>
```

Github : <https://github.com/suyashtambe/PIJ>