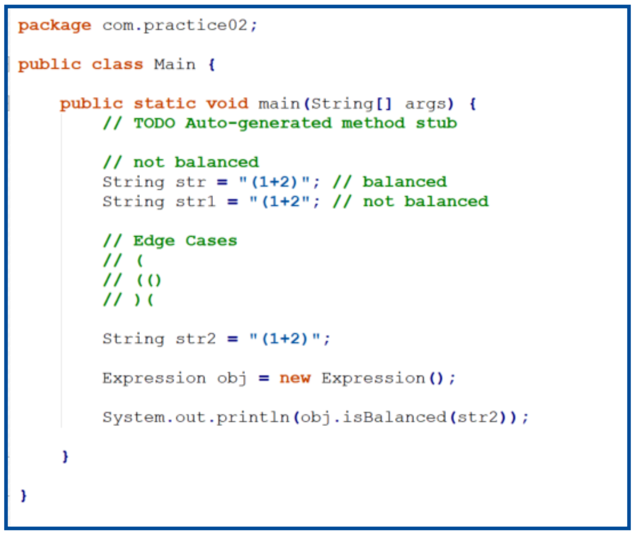
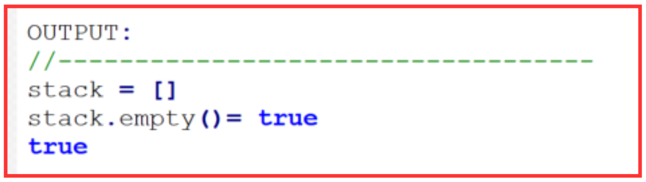
Stacks - Check Expressions

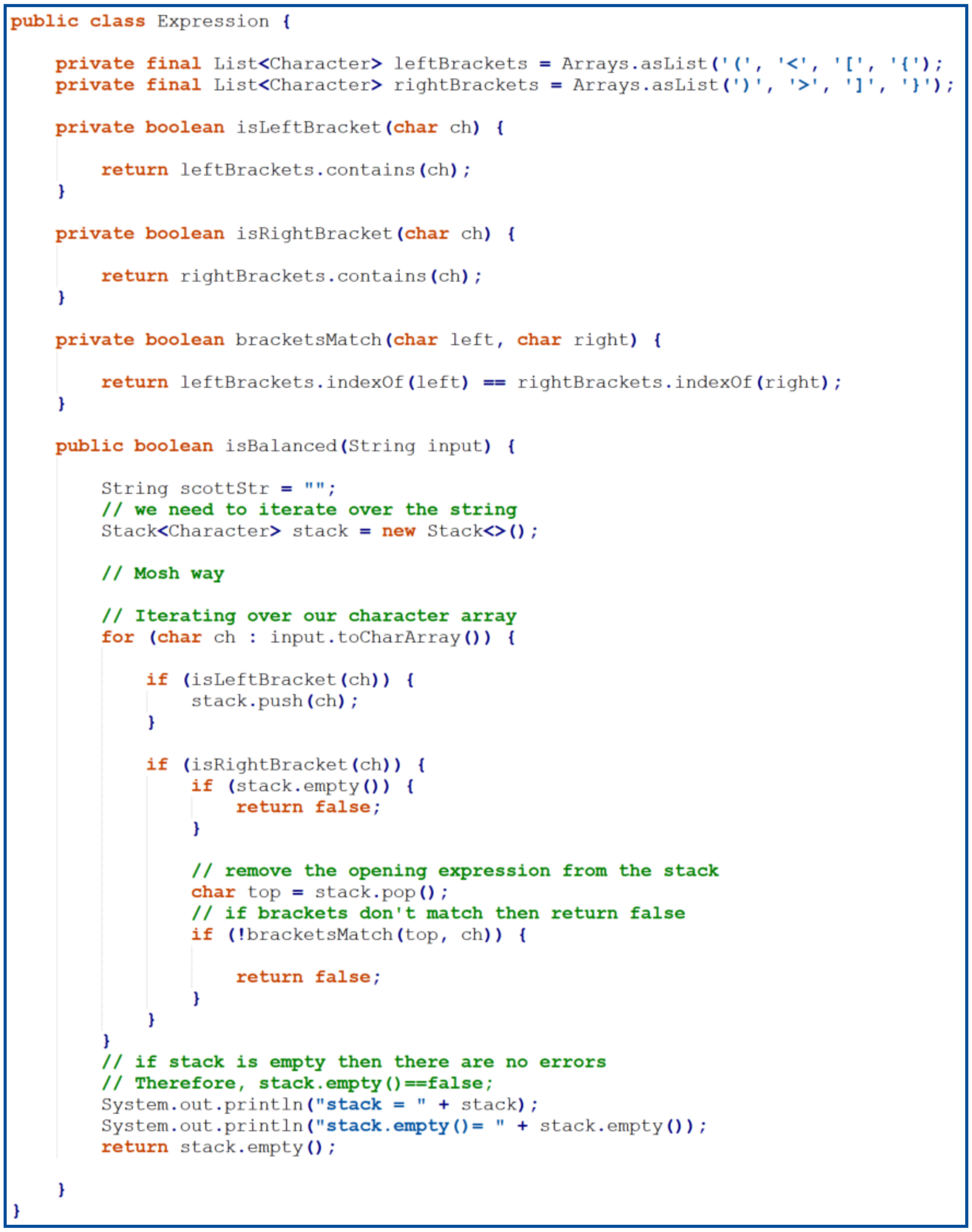


```java
package com.practice02;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // not balanced
        String str = "(1+2)"; // balanced
        String str1 = "(1+2"; // not balanced

        // Edge Cases
        // (
        // (()
        // )(

        String str2 = "(1+2)";

        Expression obj = new Expression();

        System.out.println(obj.isBalanced(str2));

    }

}
```

Output:

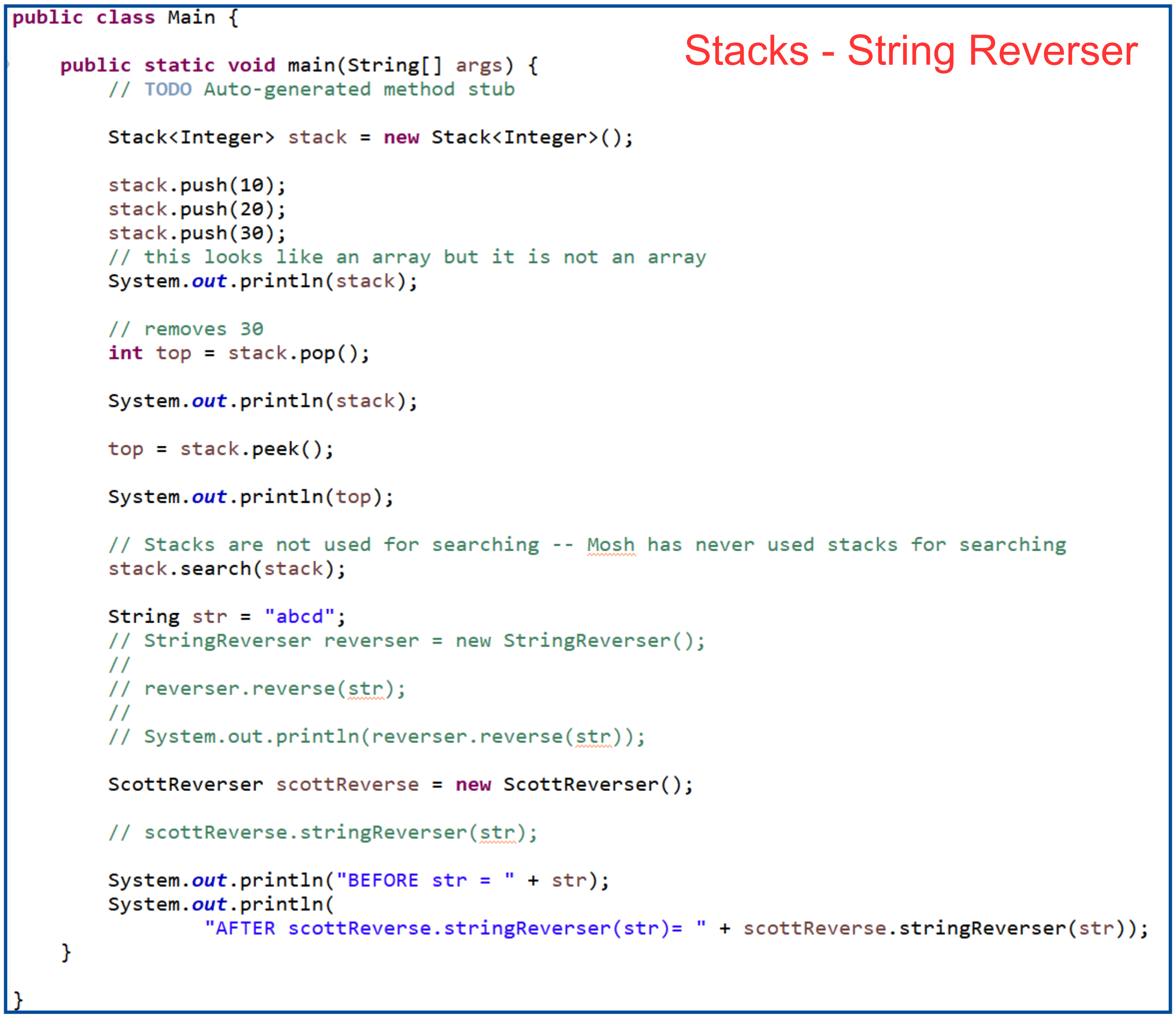


```
OUTPUT:
//-----------------------------------
stack = []
stack.empty()= true
true
```

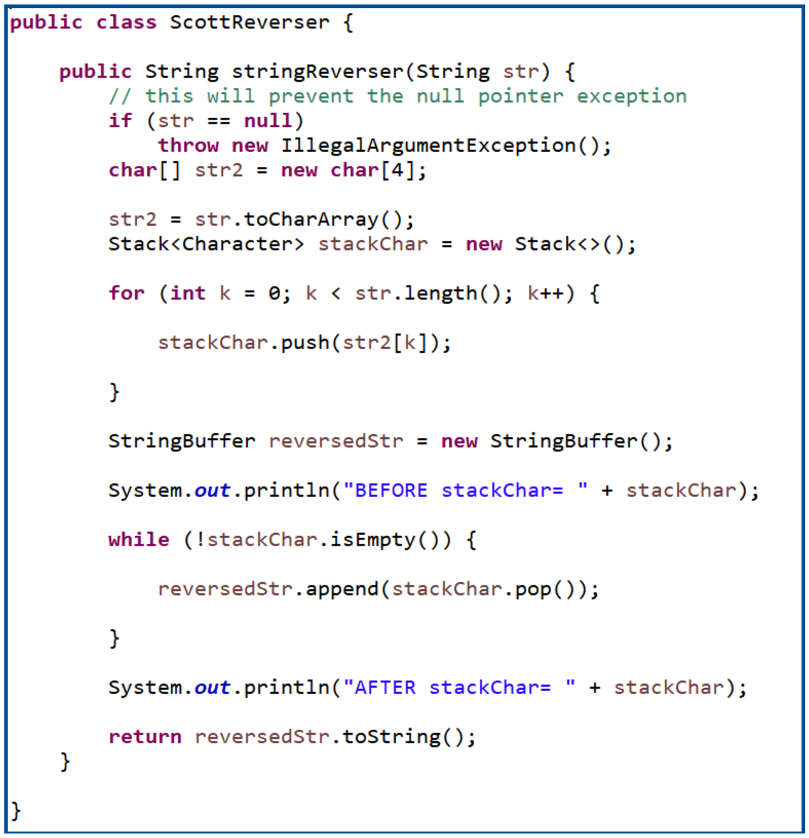


```java
public class Expression {

    private final List<Character> leftBrackets = Arrays.asList('(', '<', '[', '{');
    private final List<Character> rightBrackets = Arrays.asList(')', '>', ']', '}');

    private boolean isLeftBracket(char ch) {

        return leftBrackets.contains(ch);
    }

    private boolean isRightBracket(char ch) {

        return rightBrackets.contains(ch);
    }

    private boolean bracketsMatch(char left, char right) {

        return leftBrackets.indexOf(left) == rightBrackets.indexOf(right);
    }

    public boolean isBalanced(String input) {

        String scottStr = "";
        // we need to iterate over the string
        Stack<Character> stack = new Stack<>();

        // Mosh way

        // Iterating over our character array
        for (char ch : input.toCharArray()) {

            if (isLeftBracket(ch)) {
                stack.push(ch);
            }

            if (isRightBracket(ch)) {
                if (stack.empty()) {
                    return false;
                }

                // remove the opening expression from the stack
                char top = stack.pop();
                // if brackets don't match then return false
                if (!bracketsMatch(top, ch)) {

                    return false;
                }
            }
        }
        // if stack is empty then there are no errors
        // Therefore, stack.empty()==false;
        System.out.println("stack = " + stack);
        System.out.println("stack.empty()= " + stack.empty());
        return stack.empty();

    }
}
```

Stacks - String Reverser

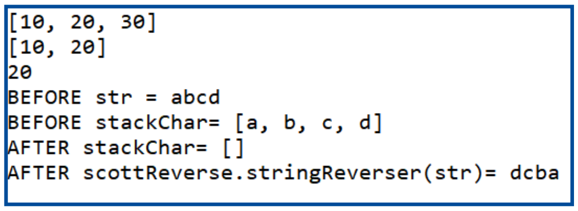


```java
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Stack<Integer> stack = new Stack<Integer>();

        stack.push(10);
        stack.push(20);
        stack.push(30);
        // this looks like an array but it is not an array
        System.out.println(stack);

        // removes 30
        int top = stack.pop();

        System.out.println(stack);

        top = stack.peek();

        System.out.println(top);

        // Stacks are not used for searching -- Mosh has never used stacks for searching
        stack.search(stack);

        String str = "abcd";
        // StringReverser reverser = new StringReverser();
        //
        // reverser.reverse(str);
        //
        // System.out.println(reverser.reverse(str));

        ScottReverser scottReverse = new ScottReverser();

        // scottReverse.stringReverser(str);

        System.out.println("BEFORE str = " + str);
        System.out.println(
                "AFTER scottReverse.stringReverser(str)= " + scottReverse.stringReverser(str));
    }

}
```

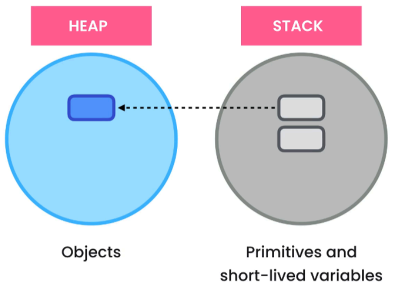


```java
public class ScottReverser {

    public String stringReverser(String str) {
        // this will prevent the null pointer exception
        if (str == null)
            throw new IllegalArgumentException();
        char[] str2 = new char[4];

        str2 = str.toCharArray();
        Stack<Character> stackChar = new Stack<>();

        for (int k = 0; k < str.length(); k++) {

            stackChar.push(str2[k]);

        }

        StringBuffer reversedStr = new StringBuffer();

        System.out.println("BEFORE stackChar= " + stackChar);

        while (!stackChar.isEmpty()) {

            reversedStr.append(stackChar.pop());

        }

        System.out.println("AFTER stackChar= " + stackChar);

        return reversedStr.toString();
    }

}
```

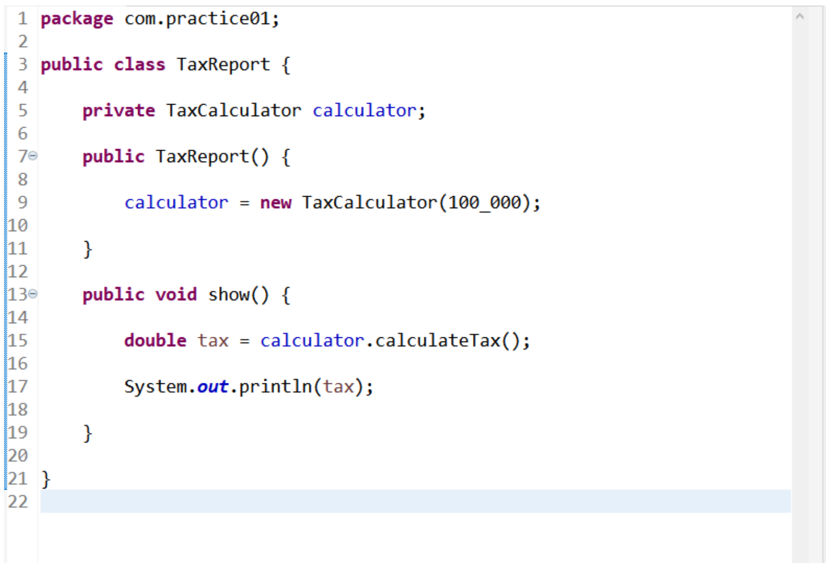
Output:



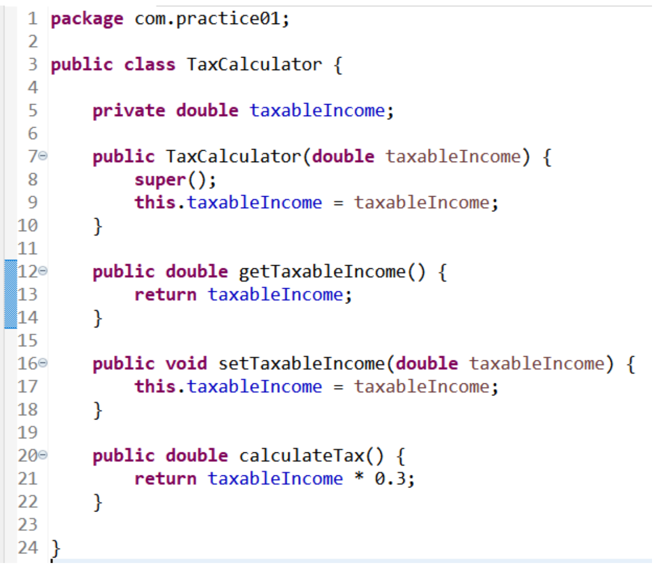
```
[10, 20, 30]
[10, 20]
20
BEFORE str = abcd
BEFORE stackChar= [a, b, c, d]
AFTER stackChar= []
AFTER scottReverse.stringReverser(str)= dcba
```



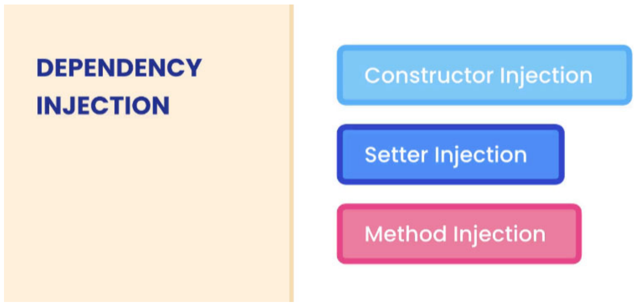
These Classes are tightly coupled and we need to use interfaces prevent tight coupling



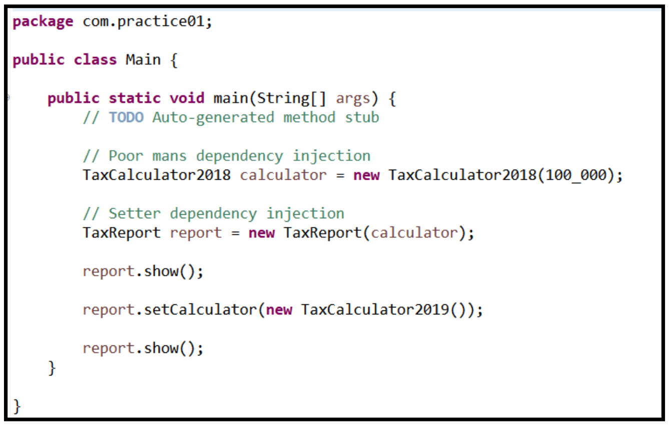
```java
package com.practice01;

public class TaxReport {

    private TaxCalculator calculator;

    public TaxReport() {

        calculator = new TaxCalculator(100_000);

    }

    public void show() {

        double tax = calculator.calculateTax();

        System.out.println(tax);

    }

}
```



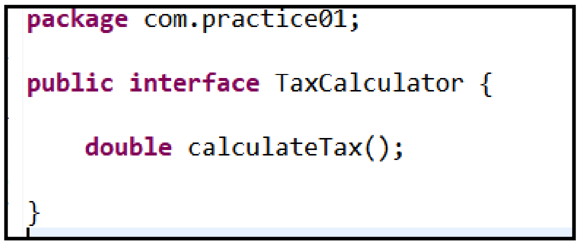
```java
package com.practice01;

public class TaxCalculator {

    private double taxableIncome;

    public TaxCalculator(double taxableIncome) {
        super();
        this.taxableIncome = taxableIncome;
    }

    public double getTaxableIncome() {
        return taxableIncome;
    }

    public void setTaxableIncome(double taxableIncome) {
        this.taxableIncome = taxableIncome;
    }

    public double calculateTax() {
        return taxableIncome * 0.3;
    }

}
```



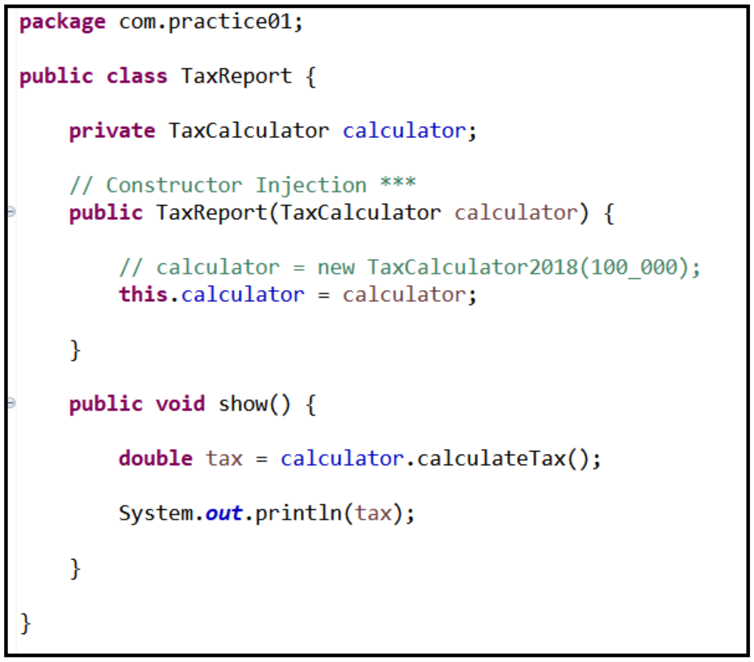
Constructor Injection



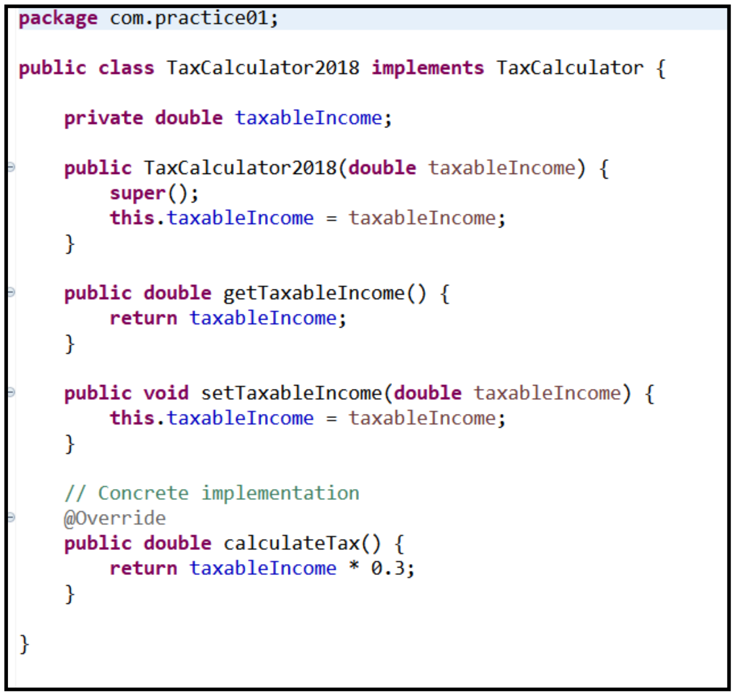
```java
package com.practice01;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // Poor mans dependency injection
        TaxCalculator2018 calculator = new TaxCalculator2018(100_000);

        // Setter dependency injection
        TaxReport report = new TaxReport(calculator);

        report.show();

        report.setCalculator(new TaxCalculator2019());

        report.show();
    }

}
```



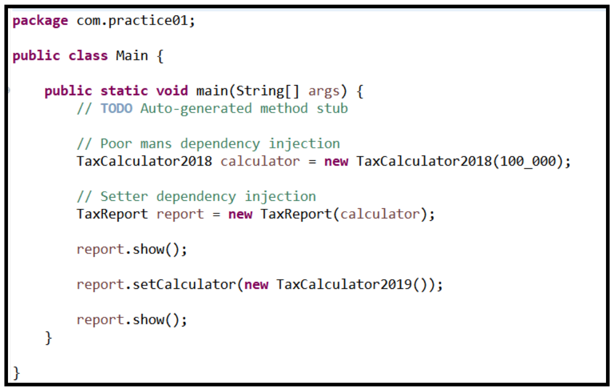
```java
package com.practice01;

public interface TaxCalculator {

    double calculateTax();

}
```



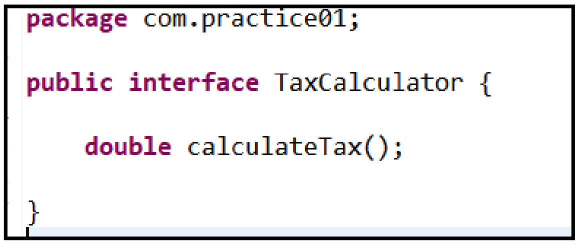
```java
package com.practice01;

public class TaxReport {

    private TaxCalculator calculator;

    // Constructor Injection ***
    public TaxReport(TaxCalculator calculator) {

        // calculator = new TaxCalculator2018(100_000);
        this.calculator = calculator;

    }

    public void show() {

        double tax = calculator.calculateTax();

        System.out.println(tax);

    }

}
```



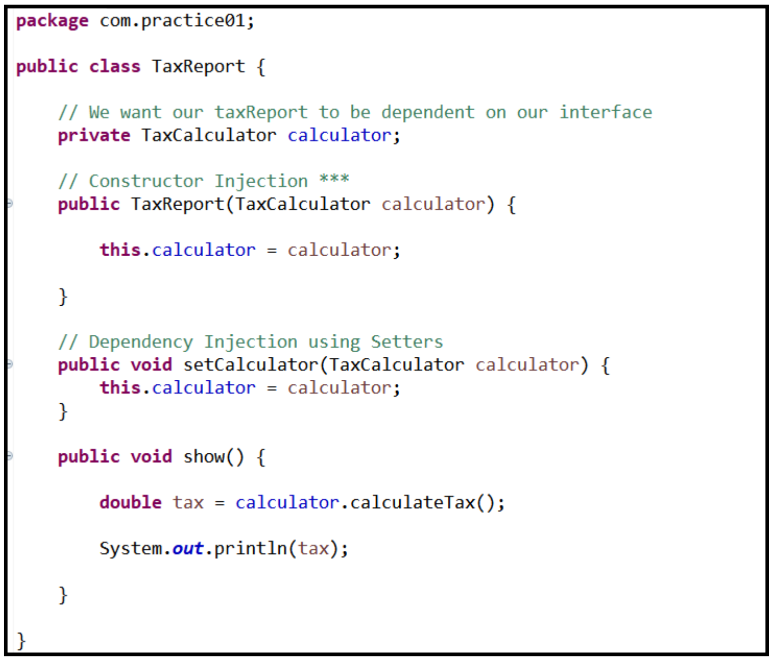
```java
package com.practice01;

public class TaxCalculator2018 implements TaxCalculator {

    private double taxableIncome;

    public TaxCalculator2018(double taxableIncome) {
        super();
        this.taxableIncome = taxableIncome;
    }

    public double getTaxableIncome() {
        return taxableIncome;
    }

    public void setTaxableIncome(double taxableIncome) {
        this.taxableIncome = taxableIncome;
    }

    // Concrete implementation
    @Override
    public double calculateTax() {
        return taxableIncome * 0.3;
    }

}
```



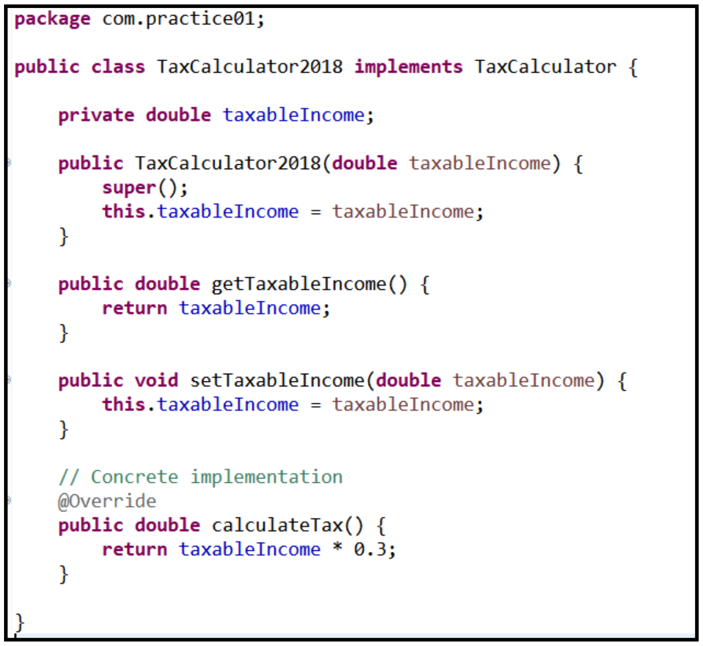
```java
package com.practice01;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // Poor mans dependency injection
        TaxCalculator2018 calculator = new TaxCalculator2018(100_000);

        // Setter dependency injection
        TaxReport report = new TaxReport(calculator);

        report.show();

        report.setCalculator(new TaxCalculator2019());

        report.show();
    }
}
```



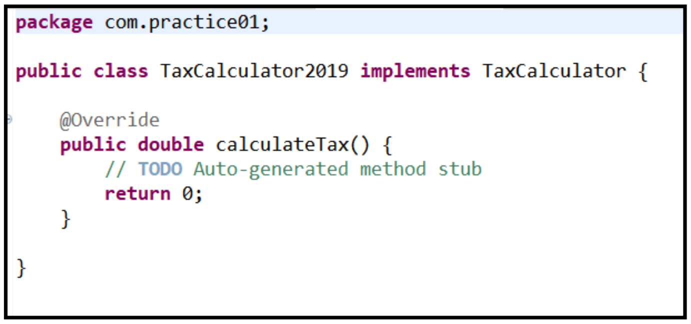
```java
package com.practice01;

public interface TaxCalculator {

    double calculateTax();

}
```



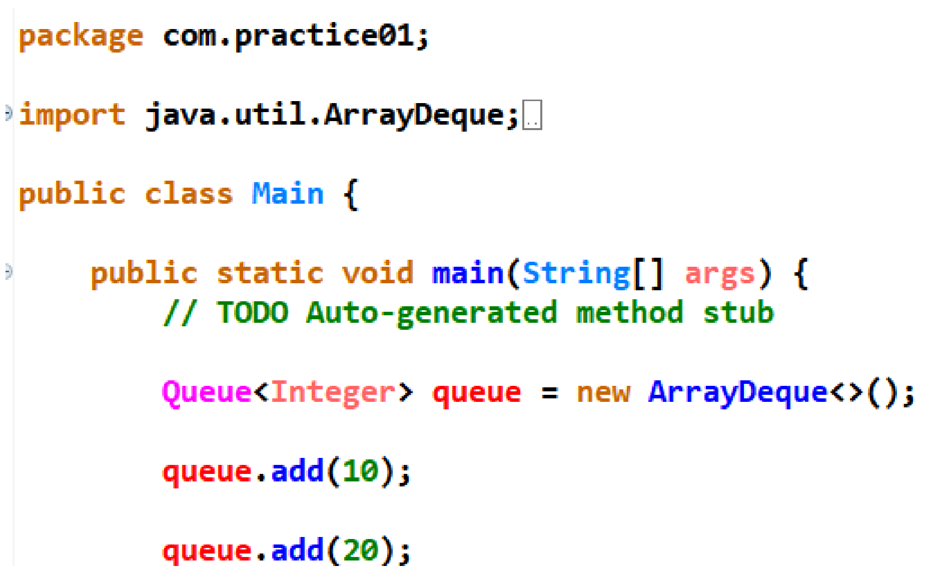
```java
package com.practice01;

public class TaxReport {

    // We want our taxReport to be dependent on our interface
    private TaxCalculator calculator;

    // Constructor Injection ***
    public TaxReport(TaxCalculator calculator) {

        this.calculator = calculator;

    }

    // Dependency Injection using Setters
    public void setCalculator(TaxCalculator calculator) {
        this.calculator = calculator;
    }

    public void show() {

        double tax = calculator.calculateTax();

        System.out.println(tax);

    }
}
```



```java
package com.practice01;

public class TaxCalculator2018 implements TaxCalculator {

    private double taxableIncome;

    public TaxCalculator2018(double taxableIncome) {
        super();
        this.taxableIncome = taxableIncome;
    }

    public double getTaxableIncome() {
        return taxableIncome;
    }

    public void setTaxableIncome(double taxableIncome) {
        this.taxableIncome = taxableIncome;
    }

    // Concrete implementation
    @Override
    public double calculateTax() {
        return taxableIncome * 0.3;
    }

}
```

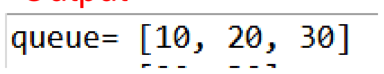


```java
package com.practice01;

public class TaxCalculator2019 implements TaxCalculator {

    @Override
    public double calculateTax() {
        // TODO Auto-generated method stub
        return 0;
    }

}
```

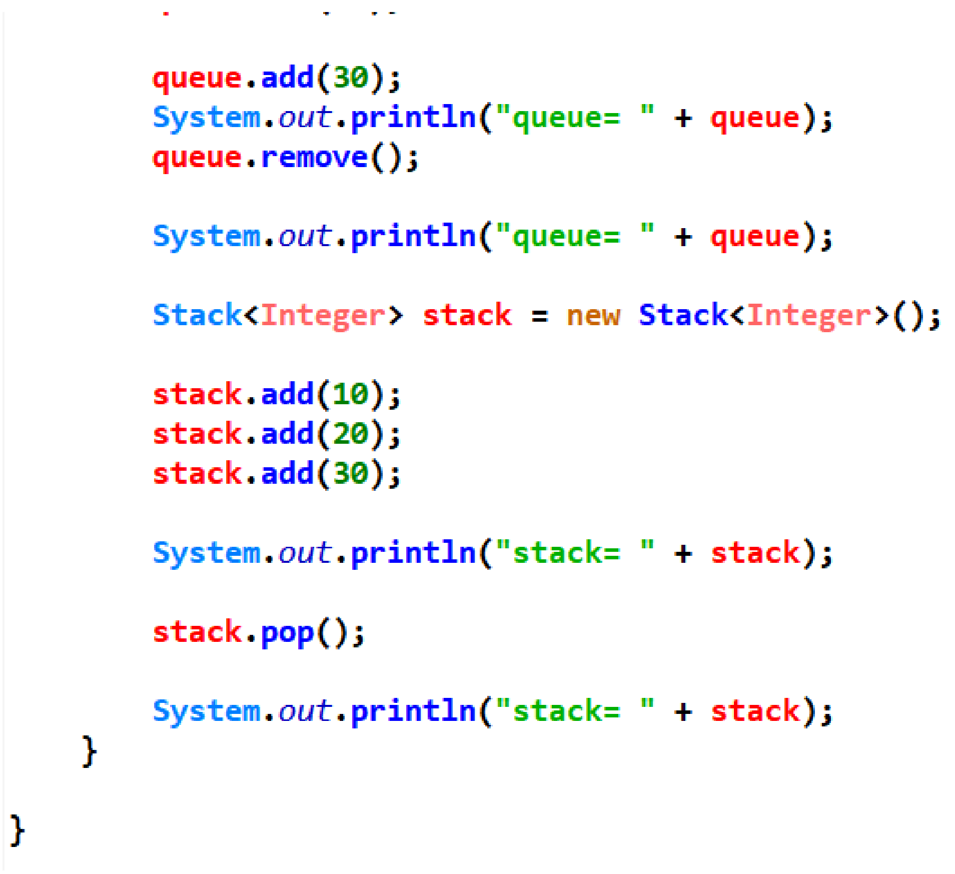


```java
package com.practice01;

import java.util.ArrayDeque;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        Queue<Integer> queue = new ArrayDeque<>();

        queue.add(10);

        queue.add(20);
```

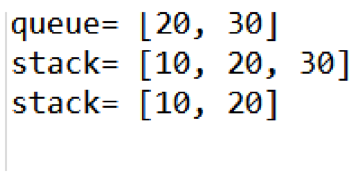
Output



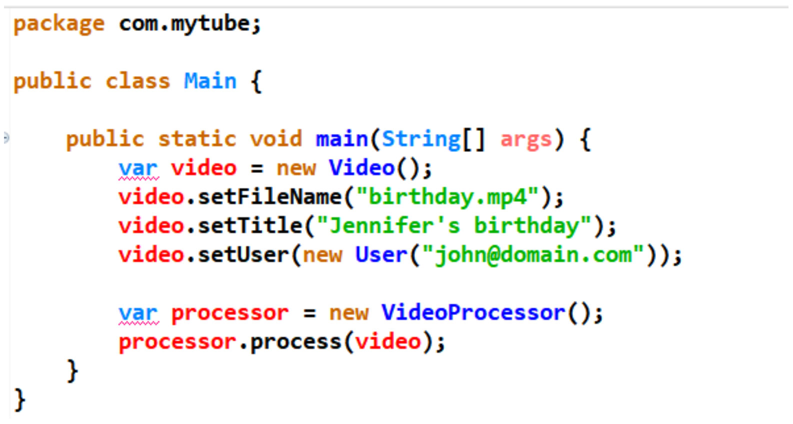
```
queue= [10, 20, 30]
```



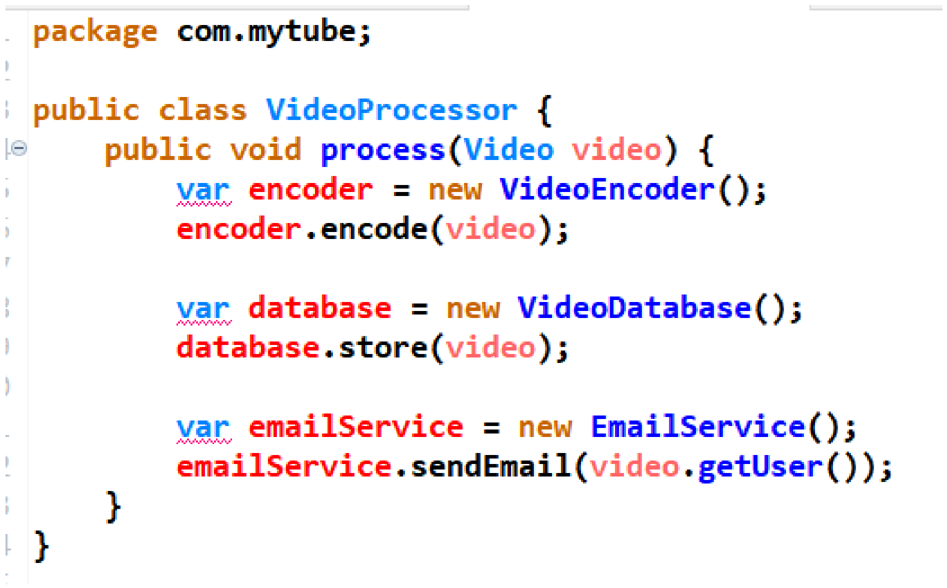
```
        queue.add(30);
        System.out.println("queue= " + queue);
        queue.remove();

        System.out.println("queue= " + queue);

        Stack<Integer> stack = new Stack<Integer>();

        stack.add(10);
        stack.add(20);
        stack.add(30);

        System.out.println("stack= " + stack);

        stack.pop();

        System.out.println("stack= " + stack);
    }

}
```



```
queue= [20, 30]
stack= [10, 20, 30]
stack= [10, 20]
```



```
package com.mytube;

public class Main {

    public static void main(String[] args) {
        var video = new Video();
        video.setFileName("birthday.mp4");
        video.setTitle("Jennifer's birthday");
        video.setUser(new User("john@domain.com"));

        var processor = new VideoProcessor();
        processor.process(video);
    }
}
```



```
package com.mytube;

public class VideoProcessor {
    public void process(Video video) {
        var encoder = new VideoEncoder();
        encoder.encode(video);

        var database = new VideoDatabase();
        database.store(video);

        var emailService = new EmailService();
        emailService.sendEmail(video.getUser());
    }
}
```