

# 23/12/22 (21) String Programming, Encapsulation, Github Eclipse integration

## Agenda

① Program - ③

- i) Palindrome
- ii) Anagram
- iii) Pangram

② Encapsulation :- Security, Private, Setter, getter  
Beam.

③ github Eclipse integration.

## Programs

① Palindrome :- A string to be a palindrome if it is the same if we start reading it from left to right (on right to left).

Input : NITIN

O/p :- "it is palindrome".

## Code :-

```
String s1 = "NITIN";  
String s2 = "";  
for (int i = s1.length() - 1; i >= 0; i--) {  
    s2 = s2 + s1.charAt(i);  
}  
if (s2.equals(s1)) {  
    S.o.p("it is palindrome");  
}  
else {  
    S.o.p("it is not Palindrome");  
}
```



② Anagram:- Two strings are said to be anagram if they make a meaningful word by rearranging or (or) shuffling the letters of the string.

Eg HEART LISTEN  
~~↓ ↓ ↓~~ ~~↓ ↓ ↓~~  
EARTH SILENT

Code:-

String s1 = "Race";

String s2 = "CarE";

// Make all letters to lowercase

s1 = s1.toLowerCase();

s2 = s2.toLowerCase();

// store them in an array

char[] ch1 = s1.toCharArray();

char[] ch2 = s2.toCharArray();

// Sort them in order

Arrays.sort(ch1);

Arrays.sort(ch2);

// Comparing two arrays of characters

if (Arrays.equals(ch1, ch2)) {

    s.o.p("It's anagram");

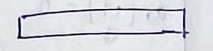
else {

    s.o.p("It's not anagram");

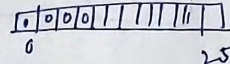
③ Pangram:- the string is called a Pangram if it contains all the alphabets from a to z (or A to Z, ignoring the case sensitivity).

Eg THE QUICK BROWN FOX JUMPS OVER LAZY DOG

Logic

s1 → 

Empty array



remove spaces before

creating array

Code:- Eg- The QUICK BROWN FOX JUMPS OVER LAZY DOG

boolean flag = false;

String s1 = "THE QUICK BROWN FOX JUMPS OVER LAZY DOG";

// Replacing Spaces

s1 = s1.replace(" ", "");

System.out.println(s1);

// Creating an array

char[] ch = s1.toCharArray();

// creating empty array from 0 to 25 index

int[] arr = new int[26];

// Moving through array ch

```
for (int i=0; i<ch.length; i++) {
```

```
    int index = ch[i] - 65;
```

```
    ar[index]++;
```

T --> 84 (ASCII)  
 $84 - 65 = 19$

ar[19] = 0

↓  
ar[19]++ = 1

// Moving through ar

```
for (int i=0; i<ar.length; i++) {
```

```
    if (ar[i] == 0) {
```

```
        S.O.P("it is not Pangram");
```

```
        flag = true;
```

```
}
```

```
if (flag == false) {
```

```
    System.out.println("it is Pangram");
```

output : it is Pangram.



① OOP's (Object-Oriented Programming System)  
it is a Methodology (or Paradigm) to design a Program using classes and objects.

→ JAVA - four pillars

- ① Encapsulation → Privacy (or Security) → Data-hiding ↓ Binding.
- ② Inheritance → Code Reusability.
- ③ Polymorphism → Code flexibility.
- ④ Abstraction → Implementation hiding.  
↓  
Only features are ~~visible~~ Visible.

① Encapsulation :- is a process of wrapping code and data together into a single unit. ~~+~~  
Examples:- a capsule which is mixed of several medicines.

Encapsulation:-

- ① data hiding
- ② Data binding
- ③ Providing security
- ④ Providing Controlled access to Data members.

→ Private, letter, getter, ~~these~~ these ~~are~~ known as shadowing.

① Class Student { data with No security

int age; // Instance Variable, Data Member, field Members, Properties.

String name;

String City;

}

Public Class Encap {

Public static void main (String[] args) {

Student st = new Student();

st.age = 28;

st.name = "Hyder";

st.City = "Bengaluru";

}

}

→ it is not good practice. There is no security for data.

→ Anyone can access the data.

→ We have provide restriction. data members should only accessed within the class.

→ for security direct access should be restricted.



① Security for data is provided by using Private keyword.

- data can't be accessed outside the class directly.
- it is accessed within the class.
- instance (or) object variable.

Code: ~~data~~ data with security.

Class Student {

```
private int age;  
private String name;  
private String city city;
```

}

public class Encap {

```
public static void main(String[] args) {
```

```
    Student st = new Student();
```

```
    st.age = 28; (X) data can't be  
                  accessed within  
                  class
```

}

→ Error

② Getter and Setter in Java

- if Private keyword is used not one can access the data ~~from~~ outside the class.
- Variable are created store data. data should be given by someone outside class.
- the data taken from outside the class and it is given to variables.
- we have control over data.

① Setter - if a method is relieving data from outside and set it to data member. Such method we called as setter.

- setting value (or) data to variable
- Setter sets (or) updates the value (or) (Mutators). it sets the value for any variable used in a class's programs.

② Getter - Getter returns the value (accessors), it returns the value of data type int, String, double, float etc.

- getter starts with word "get" followed by variable name
- setter starts with word "set" followed by variable name.



→ following Syntax is highly recommended but not mandatory.

⇒ setter Will not return anything.

→ it receives data and set to data Member (or) Variable

→ getter <sup>must</sup> not accept any input.

Code:-

Class Student1 {

Private int age;

Private String name;

Private String City;

~~Private~~

1. // Setting age and returning it

// Setters and getter of data Members

Void SetAge(int a) {

age = a;

}

int getAge() {

return age;

}

} Method to take age and return it.

Void setName(String n) {

name = n;

}

String getName() {

return name;

}

} Method to take name & return it.

Void setCity(String c) {

City = c;

}

String getCity() {

return City;

}

} Method to take City and return it.

Public Class Encap {

Public void main(String[] args) {

Student1 std = new Student1();

① ⇒ std.setAge(23); → giving age to var  
int age = std.getAge(); → invoking age.

S.O.P(age) // output: 23

② ⇒ std.setName("ashish");  
String name = std.getName();

S.O.P(name)  
std.setCity("Hyd");  
String City = std.getCity();  
y y



→ Encapsulation refers to providing controlled access to the data member of the class by avoiding (or) preventing direct access.

→ Encapsulation provides data binding.

① How to achieve Encapsulation in Java?  
by using Private Members, setters and getters.

(ii) Beam: A class which has all the data members as Private is called beam.

Eg. 

```
class student {  
    private int age;  
    private String name;  
    private String City;  
}
```

(iii) Shadowing

It is a Computer programming phenomenon in which a variable declared in one scope (like decision block, Method (or) inner class) has the same name as another declaration of the enclosing scope.

→ When ever there is naming conflict between instance variable and local variable ~~of method~~ within setter.

Eg. 

```
class student {  
    private int age; // local Variable  
    void setAge(int age) {  
        age = age; // instance Variable  
    }  
}
```

```
public class Encapsulation {  
    public static void main(Strings[] args) {  
        student std = new student();  
        std.setAge(23);  
        int age = std.getAge();  
        S.o.p(age) → o/p: 0  
    }  
}
```

→ JVM will not allocate any Value.  
for string it gives null  
for int it gives '0' (Zero).

### ① Resolving Shadowing Problem :-

→ this problem is resolved by using 'this' keyword.

Eg - class Student {

private int age;

private ~~int~~ String name;

⇒ void setAge(int age) {

**this**.age = age

}

int getAge() {

return age;

}

⇒ void setName(String name) {

**this**.name = name;

}

int getName() {

return name;

}}