Chapter 16

Variables in detail

What is a variable?

1. A Variable **is like a container**, which **holds some data**
2. The **container(variable)** will have **specific size(300g)** associated with it, so that we can **store maximum that much data**

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

- Variable is **used to store data**
- Every **variable is assigned a data type** that describes the **type** and **quantity** of value it can hold
- Value stored in a variable can be **changed anytime** during program execution

Variable declaration

type variablename;

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

- **Variables must have a <u>type</u>**
- **Variables must have a <u>name</u>**

type variablename;

Primitive type

1. byte
2. short
3. int
4. long
5. double
6. float
7. boolean
8. char

Reference type

1. class
2. interface
3. array
4. enum

primitive variables

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

reference variables

```
class Student{
    String name;
    String studyClass;
    int rollno;
    double percentage;
    House h;
```

Is String a primitive type or Reference type?

String is a class, so it is a reference type

Rules to name a variable

1. **Allowed characters** are:
    1. Alphanumeric characters [A-Z], [a-z], [0-9]
    2. $ dollar sign
    3. _ underscore
2. **Should not start with digit**
3. Names are **case-sensitive**
4. Name **should not be a keyword**

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

Rules of an identifier are applicable to name a variable, class and a method

What happens when you declare a variable?

Local variables doesn't get a default value

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

Initializing a value to a variable

variablename = literal/expression

suitcase1 = 1000 (literal)

suitcase1 = suitcase2+suitcase3

suitcase1 = 1000
5000

name = null
studyClass = null
rollno = 0
percentage = 0.0

21332738

Student s = new Student();

Memory

- **Primitive variables hold actual value**
- **Reference variables hold reference to the object but not the actual object**

We can assign a value to a variable any number of times but when we assign a new value to a variable old value will be overwritten

Difference b/w Initialization & Assignment

- When we **assign a value to a variable for the first time**, it is known as variable **initialization**

- If we want to **update** it's value then we will assign a new value to it, which is known as **assignment**

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

Assigning a value to a variable

Can we declare and initialize variable at once?

- Yes, 100%

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

Assigning a value to a variable

Task

- Create a class named **VariablesDemo**
- Provide main method
- Display **Variables Demo** message as a first line of output
- Create four variables namely **a, b, c, d of type int** inside the main method
- Initialize **10 to a**, **20 to b** and **-20 to c**
- Display the **values a, b, c in three different lines**
- Display value of d

Initialize 1000 to d and run the program again

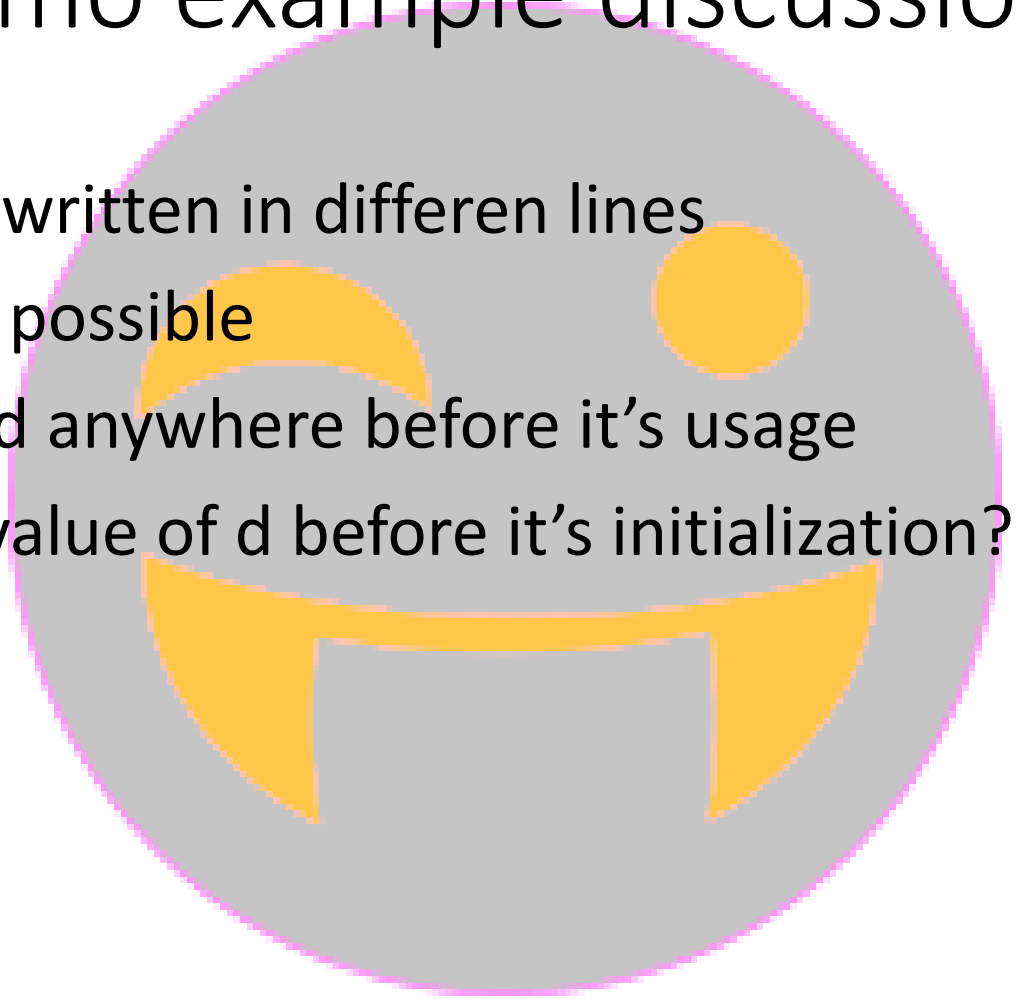
NOTE: Local variables should be assigned with a value before used

```java
class VariablesDemo{

    public static void main(String[] args){
        System.out.println("Variables Demo");
        int a, b, c, d;
        a = 10;
        b = 20;
        c = -20;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        d = 1000;
        System.out.println(d);
    }
}
```

Variables demo example discussion

- int a,b,c,d can be written in differen lines
- a=10;b=20 is also possible
- d can be initialized anywhere before it’s usage
- What if we print value of d before it’s initialization?



Types/kinds of variables

- **Local** variables
- **Instance** variables
- **Static** variables

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

```
class Student{
    String name;
    String studyClass;
    int rollno;
    double percentage;
    House h;
```

Method, constructor, block

```
void setName(String name){
    this.name = name;
}
```

```
String getName(){
    return name;
}
```

```
Student(){
    System.out.println("Student constructor");
}
```

```
{
    System.out.println("I am block");
}
```

Local variables

Local variables does not include any access modifiers such as public, private, protected etc

- Variables that are **declared within a method, constructor or block** are called local variables
- Local variables **doesn't get a default value**
- Local variables **should always have a value before** it's **usage**
- Local variables **can't be accessed outside of it's method**

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
    /*Below code is used to add three numbers, here I a
    and I have assigned three values to those variable
    /*System.out.println(suitcase1+suitcase2+suitcase3)
    System.out.println(suitcase1);
    System.out.println(suitcase2);*/
    //System.out.println("suitcase3");
    suitcase1 = 500;
    System.out.println(suitcase1+suitcase2+suitcase3);
```

They are usually created when we enter a method or constructor and are destroyed after exiting the block or when the call returns from the method

Overview of access modifiers

4 Access modifiers

1. Public
2. Private
3. protected
4. default (no keyword)

An access specifier can be given to a method, variable or a class

Overview of access modifiers

- Epudu chudandi, mana entlo unde anni vishayalani manam bayata pettam
- Dabbulu ekkada dastamu, Bangaram ekkada dastamu. Elanti vishayalu anni secret ga evarikee teliyakunda unchutamu
- Epudu andaru home tours chestunnaru kabatti, konni vishayalu telustunnai🤣🤣. Manam kuda chesamu…😄😄
- Alane programming lo kuda konni vishayalani, andarkiee kanipinchela, konni vishayalani kontamandike kanipinchela cheyagalam…anduke access modifiers anevi use chestamu….avi total ga 4 untai…
- public, private, protected
- Adenti 4 ani cheppi, 3 chupincharu? Okavela em use cheyakapothe default access modifier

You already know one access specifier

```
public static void main(String[] args){
    System.out.println("Welcome to suresh techs, I am learning Java.");
    System.out.println("My name is suresh, I will get job soon");
    System.out.println(1);
    System.out.println(2);
    System.out.println(3);
    System.out.println(4);
    System.out.println("\"Suresh techs\" is 5 star");
}
```

An access specifier can be given to a <u>method</u>, variable(<u>not to a local variable</u>) or a class

public

Local variables does not include any access modifiers such as public, private, protected etc

```
public static void main(String[] args){
    System.out.println("Welcome to suresh techs, I am learning Java.");
    System.out.println("My name is suresh, I will get job soon");
    System.out.println(1);
    System.out.println(2);
    System.out.println(3);
    System.out.println(4);
    System.out.println("\"Suresh techs\" is 5 star");
}
```

- classes, methods, or variables that are **declared as public** are **accessible from everywhere in the application**
- It is specified using the keyword **public**
- There is a lot to discuss, will discuss in access modifiers chapter

Instance variables

Instance variables include access modifiers such as public, private, protected etc

- Variables that are declared **within the body of the class but outside of a method, constructor, or block** are called **Instance** variables
- Instance variables **gets a default value**
- These variables are **created when we create an object** and are **destroyed when the object is destroyed**
- Each and **every object** will have it's **own copy of instance variables**
- Instance variables **can be accessed any where in the class**

```
class Student{
    String name;
    String studyClass;
    int rollno;
    double percentage;
    House h;

//first student
Student s = new Student();

//Second student
Student s1 = new Student();
```

Task 2

- Create an **instance variable** named **marks** of **type int**
- Add values inside **a and b** and **store the result** in **new variable** called **abSum**
- Assign value of **abSum** in c
- Display values of **a, b and c in three different lines (tell me the answer)**
- Create **a local reference variable** named student1 of type **Student**
- Display value of student1**(tell me the result)**
- Assign **null** value to **student1 reference variable** before displaying

Not possible

```java
class VariablesDemo{

    int marks;
    int abSum = a+b;
    public static void main(String[] args){
        System.out.println("Variables Demo");
        int a, b, c, d;
```

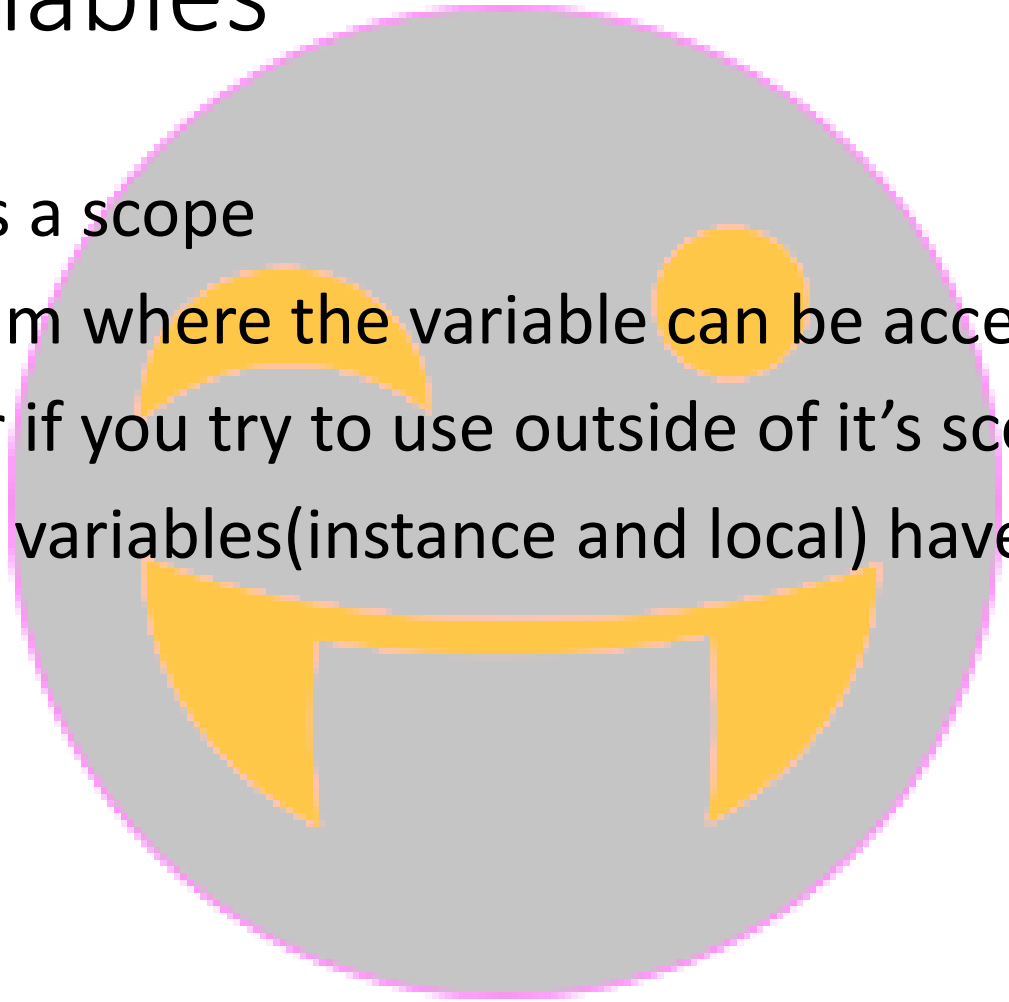
```java
class VariablesDemo{
    int marks;
    public static void main(String[] args){
        System.out.println("Variables Demo");
        int a, b, c, d;
        a = 10;
        b = 20;
        c = -20;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        d = 1000;
        System.out.println(d);
        int abSum = a+b;
        c = abSum;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

Assignment to instance variables inside a class is not possible

```
class VariablesDemo{
    int marks;
    marks = 200;
    public static void main(String[] args){
        System.out.println("Variables Demo");
        int a, b, c, d;
        a = 10;
```

Scope of variables

- Every variable has a scope
- Part of the program where the variable can be accessed/used
- Compilation error if you try to use outside of it’s scope
- Different types of variables(instance and local) have different scopes



Local Variables

- Variables that are **declared within a method, constructor or block** are called local variables
- Local variables **doesn't get a default value**
- It is **mandatory to initialize** local variables before use
- It **does not include any access modifiers** such as public, private, protected

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
```

Instance Variables

- Variables that are declared **within the body of the class but outside of a method, constructor, or block** are called **Instance** variables
- Instance variables **gets a default value**
- It is **not mandatory to initialize** instance variables before use
- It **includes access modifiers** such as public, private, protected

```
class Student{
    String name;
    String studyClass;
    int rollno;
    double percentage;
    House h;
```

Naming conventions

- It is **not mandatory to use** these rules **but many developers follow it** and it is **good practice to do so** as it will be easier for new developers to read/understand the code

- Variables & method names are written in **lower case letters**

- If there are **multiple words** in a variable name, then use the **camel case**

- **Camel case:** First letter of each word should be capitalized except first word

- Ex: collegeName, numberOfStudents

Local Variables

- Variables that are **declared within a method, constructor or block** are called local variables
- Local variables **doesn't get a default value**
- It is **mandatory to initialize** local variables before use
- It **does not include any access modifiers** such as public, private, protected

```
public static void main(String[] args){
    int suitcase1;
    int suitcase2 = 2000;
    int suitcase3 = 3000;
    suitcase1 = 1000;
```

Instance Variables

- Variables that are declared **within the body of the class but outside of a method, constructor, or block** are called **Instance** variables
- Instance variables **gets a default value**
- It is **not mandatory to initialize** instance variables before use
- It **includes access modifiers** such as public, private, protected

```
class Student{
    String name;
    String studyClass;
    int rollno;
    double percentage;
    House h;
```

Static variables

- Need to understand about **static** keyword

What next?

Static keyword

చిన్న బ్రేక్ చిటికలో వచ్చేస్తా