CS 113 Notes Midterm 1

Char stores Aski code not the actual character.

```
char x = 'a';
Sout (x);
= 97
```

Bit Permutations

1 Bit = 2^1 items
2 Bit = 2^2 items
3 Bit = 2^3 items
" " " " "

when Pinting or doing Something

("x:" +25); → x:25
("y:" + (15+50); → 65
("z:" + 300 +50); → z:30050

this sets the priority now that the object inside has priority Java skips the String y and solves the thing inside the () Since there is no string inside () it adds those two numbers

In this case no priority is set as a result Java reads the String first it goes from left to right. It concats.

if we had

(1+2 + "z"); → 3 z

Since Java runs from left to right Java does not see the string z first it adds the two numbers and then concats it.

Java identifiers (variables) cannot begin with a number, cannot contain #, cannot have & . ✓

compiler is a software that translates source code into a specific target language. (Machine language)
Machine language the lowest level language.

Java compiler translates Java source code to byte code. Java virtual Machine executes byte code

an object is defined by a class
class is a blueprint of the object
methods are behaviors that a class has
Every String in Java is an Object of the String class.
The String objects can be created in two ways:

String myString = "Hello world";
String myString = new String("Hello world"); // ✓

String can be combined using the + operator
the + operator does the task of concating and also adding.

Escape characters

("I Said \"Hello\" to you");
Hamesha pehle aata hai \.
"ke

\b backspace
\t tab
\n new line
\\ back slash

variable types

int x = 7; can be changed by x= something int type
but cannot be changed by
✗ int x = 8; ✗

CS 113 Notes Midterm 1

f you want variable to not change its value use the final keyword

final int MIN = 69;

The value of MIN cannot be changed now.

Primitive data type

For intigers;

byte, short, int, long

For floating Point;

float, double.

For chars;

char

For boolean values;

boolean.

+ addition

- subtraction

* multi

/ Division

% remainder

int / int = int

double / double = double

int / double or double / int = double

in division if the value data type is double and int division is done

ex: double x = (14/7);

the output will be 2.0

ex: double x = (7/14);

0.0

order of opreation Java

→ Parenthesis ( )

→ Exponents

→ Multiplication, division, Modulo (left to right)

→ Addition subtraction (left to right)

Assignment opreator

+=

-=

*=

/=

%=

data converson

int x = 20;

double

casting

int x = 50;

Float result = (Float) x

input

Scanner scanner = new Scanner(System.in);

Scanner.nextLine();

.nextDouble();

.nextInt();

"new" opreator is used to create an object

creating an object is called instantiation

An object is an instance of a class

The '.' opreator can be used to evoke any method a class has.

ex:

Class MyobJ = new class();

myobJ.method();

CS 113 Notes Midterm 1

A primitive variable contains the value itself.

An object variable contains the address of the object.

Primitive data types:

```
int num = 76;
```

num Holds the value of 76.

If an opreation is conducted ex:

```
int num2 = num;
```

num2 Holds 76.

This will now Hold 76;

For refrence data type:

```
String Str = "Hello";
```

Str Holds the address to where "Hello" is stored.

if an opreation is conducted ex:

```
String Str2 = Str;
```

Now this Holds the address to "Hello"

this Holds the address to "Hello"

Strings are immutable

commonly used String methods

```
String Str = "Hello world";
```

Str.toUpperCase(); → Makes everything upper case

Str.concat("."); → adds . at the end of the String

Str.replace('E','x'); → Replaces every E with x in the String

Str.subString(3,10); → Returns String that starts from 3 index and ends at index 10-1.

subString method creates a new object in memory

String Direct assignment vs object creation

when a String object is created with the new keyword;

```
String Str1 = new String();
String Str2 = new String();
```

regardless of whats inside the variables Str1 & Str2 Store different address and Str1 == Str2 can never be true

when the String is created using direct assignment;

```
String x = "Hello";
String y = "yes";
String z = "Hello";
```

when two String variables refer to the same string in this case "Hello" the store the same address in their variable and hence x==z = True

However when a String variable is assigned to a onique value the variable stores a onique address.

Example of aliases

Aliases

Two or more refrences that refer to the same object

Random class

```
Random random = new Random();
```

For random Int use

random.nextInt(7)-2; → Range [-2, 5)

random.nextInt(10)-5; → -5 to 4 ← imp

Math class always returns a double

no need to create object.

Math.Pow(7,2); → 7^2

Math.Sqrt(x);

Math.cos(90);

Math.random()*

Math.random()*5+1;

where it ends

where it starts

CS 113 Notes Midterm 1

Methods

calling a method is called invoking

Method can be written with or without peremeters.

Methods may or may not return anything if it retons it should have a return type if not it should be void.

```
Public int (int z) {
    do this
    return int
}
```

varables declared inside a method are local variables and cannot be accessed outside that method they are destroyed.

object has state and behavior

constructor is used to setup an object when it is initially created

Constructor has same name as class. and does not have a return type not even void

```
Public Constructor () {

}
```

Variable declared at class leavel is called instance data

when an instance of a class is creatd (object) it has its own instance data variable.

objects share methods bot have their own data space.

visibility modifiers (encapsolation)

final, Pubic, protected, private

→ mempers of the class with pobric visibility can be used any cohere

→ members declared with privdle can only used within that class

→ members declared without can be accessed anywhere in the same package

- default constructor does not have to be written it is automatic and accepts no peremeters.

→ there can be multiple ~~classes~~ constructors in a class with dift peremeter types.

== equal to

!= not equal to

< less than

> greater than

<= less than or equal to

>= greater than or equal to

if statement Format

```
if (condition){
    do this;
}
```

If the if Statement only has one line you do not need the {}.

logical opreator

|| OR

&& and

! not

! not opreator inverses the Boolean value

! false = true

! true = false

while statements

```
while (condition) {
    statement
}
```

```
if (condition) {
    statement;
}
else {
    statement;
}
```

It is recomended to not use == sign to compare a double or float. It is recomended to use Tolerance do:

```
if (Math.abs(f1-f2) < TOLERANCE)
```

→ Tolerance is a variable with a double value.

unicode

0-9	48 – 57
A-Z	65– 90
a-z	97– 122

The .equals() {} can be used to find if two strings are exactly the same.

The .compareTo () method can be used on two string values.

this returns a possitive number if S1 > S2

" " " negative " " S1 < S2

" " 0 if S1 = S2

this checks each individual index if one has more chars than S2 more index ex S2 has more chars returns possitive

```
S1.compareTo(S2);
S1 = "Hello"
S2 = " Hello"
S3 = "hello"
```

ArrayList

```
ArrayList<datatype> x = new ArrayList<datatype>();
```

→ cannot store primitive data type but can store wrapper classes.

```
x.add( );
x.add(index, Fobs);
x.remove(int index);
x.get(index);
x.isEmpty();
x.size();
```

⑤ .equals();

Method checks every character in the string and returns true if everything is the same.

6) switch statement

```
switch (option){
  case "A":
     do this;
     break;
  case "B":
  }
```

```
  default:
     do this;
```

switch cannot be used for floating point values

while loop

```
while (condition) {
    statement;
    increment;
}
```

for each loop

```
for ( datatype variable : arrayListVariable ) {
    sout (variable);
}
```

conditional operator

```
larger = (num1 > num2) ? num1 : num2);
```

(num1 > num2) → checks the condition; ? → if its true it does this; num1 → this; : num2 → else this

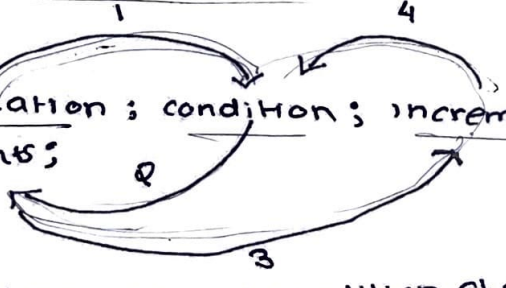
do while loop format

```
do {
  statements;
} while (condition)
```

do while loop always runs once.

for loop

```
for (initialization ; condition ; increment) {
    statements;
}
```



order initialization → condition check → statement exe → increment

Static Methods and variables can be invoked through the class name.

Example (Different classes)

```
public class StaticVarClass {
    public static int x = 0;
    public static void method(){
        System.out.Println("Hello");
    }
}

public class Oof {
    public static void main (String[]args){
    StaticVarClass.x;
    StaticVarClass.method();
}
}
```

[when in different classes class name.method() or class name.var] is required to access method or variable that's static.

(Same class)

```
Pubic class MyClass {
    public static int z = 100;
    pubic static int method (int z){
        return(z);
    }
    public static void main (String[] args){
        method(7);
        sout(z);
    }
}
```

[when in same class you dont need the class name]

Practice exam

All the objects of the class that has a static method or variable. share the variable so if the variable is changed by one object it is also changed for another object (only a single copy exists)

when declaring a variable or a method the modifiers or keywords they can be in a different order ex:

```
static public int x = 7;
```

but it is prefered to have public (convention visibility modifier to come first).

Class Relationships

Dependency A uses B
(Method call or parameter)
lets say a class called library exists

this class might use another class B (example findAvalableBook that checks which books are avalable.
in this case the Library class is depended on B class
A uses B but B does not depend on A.

Aggregation A has-a B
(one class contains instance of another class)

Aggigration object contains refrences to other objects as instance data.

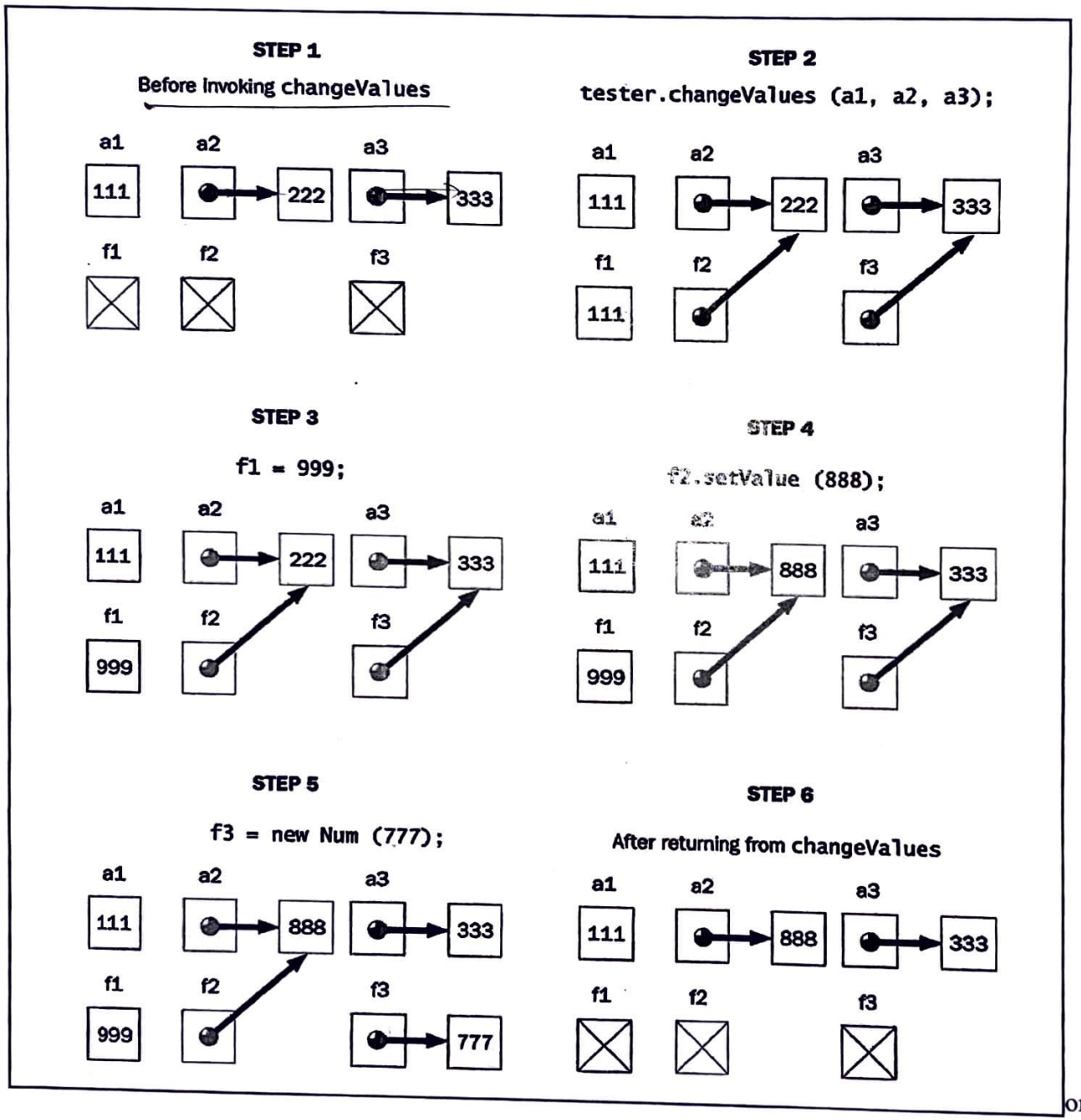
Memory space for static variable is created when the class is first refrenced.

static methods cannot refrence instance variables because instance variables don't exist until an object of that class is created.

static method can access static variables / Methods.

Method over loading

when we have methods with same name but different parameters.



CS 113 midterm 2

Array can store multiple values of the same data type.
it can be object or primitive

in java arrays itself is an object and its variable stores an address.

Declaring Arrays

```
int[] scores = new int[10];
```

- int[] — data type
- scores — Name of Array
- new int[10] — Number of elements in an array

last index of an array = number of elements − 1

Arrays can also be declared as follows

```
boolean[] x ;
x = new boolean[20];
```

Accessing elements in an array using its index

```
variable[index];
```

Assigning ~~variable~~ value at certain index

```
variable[index] = {new value};
```

changing value at certain index

```
variable[index] = variable[index] + something;
```

For each loop

```
for(int x : ArrayName)
    System.out.println(x);
```

overloaded Method can return different data type.

6.

CS 113 midterm 2

For loop to access elements of the Array

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

}

Initilizing an array at the beginning

int[] ArrayName = { 100, 720, 300, 520} ; (can only be used when declaring an array)

Arrays can be passed as a parameter

Similar to objects an refrence (address) to to the array is passed so they will be aliases of each other.

This means changing an array element inside a method will change the original

The default value of Refrence data type in an Array is Null.

" " " " primitive " " " " " " 0 for int, double, float and False for boolean

visible length Parameter list.

when you don't know the size of parameters that need to be passed into a method we can use the ...

How to Have variable length paramer

public int nums (int ... numbers) { (variable)

//code here

}

you can pass multiple variables as follows:

```
nums(1,2,3,4,5);
```

or

```
int[] number = {1,2,3,4,5};
nums(number);
```

when you pass something into the function it puts the values inside a list and you can access the parameters as follows:

```
Pubic static int sumNums(int... nombers) {
    int total = 0;
    for (int i : numbers) {
      total += nums;
    }
    return total
}
```

(data type: int; variable: nombers)

you can also use (numbers.. length) to check how many elements exist.

Method that accept variable length parameter can also accept other parameters but the other parameter should come before the variable lengh parameter.

[you can only have one set of variable parameters]

A Method constructor can be used to setup and accept the paremeters

```
Pubic class Local {
    Priuale string[] nums;
    Pubic Local(String... nomes) {
        Nums = names;
}   }
```

Two dimention Arrays

Two dimentional arrays can be defined as arrays of array.

Declaring 2D Arrays

```
int [][] x = new int [12][10]
```

- `int [][]` — data type
- `x` — variable
- `new int` — data type
- `[12]` — rows
- `[10]` — colomb

How to access 2D Array

```
x [Row][Colomb];
```

→ for loop to itterate over 2D Arrays

```
for(int i=0; i< x.length; i++) {
    for (int j =0, j < x[i]. length ; j++) {
        //code here
    }
}
```

initilizing 2D Array manually

```
int [] x = { {3,4,5,6}, {7,6,1,2}, {1,7,3,2} }
```

- `{3,4,5,6}` — Row 1
- `{7,6,1,2}` — Row 2
- `{1,7,3,2}` — Row 3

Recursion

→ Recursion occurs in a method and the method calles itself from its own body.

→ Recursive method has a base case where execution ends.

→ Example of Recursion Method

```
Public int sum(int num) {
    int result;
    if (num == 1) {
        result = 1;
    }
    else {
        result = num + sum(n-1);
    }
    return result;
}
```

→ Indirect recursion is when method 1 calles method 2 and method 2 calles method 1.

Inheritance

→ inheritance is used to create organized reusable classes

→ inheritance allows to derive a new class from an existing one

→ we can use the extends keyword to establish a child class →

lets say we have a class

parent class. → Public class school {

    // more code

}

child class ↓

Public class classroom extends school {

    // more code

}

→ Protected modifier is a visibility modifier is often used in Parent classes and can be used as other access modifiers like - Pobic and Priuale.

→ Protected modifier can be used when one wants a certain method or variable to be accessable to the child class but noone else.

→ only public, Protected methods and variable are inherited by the child class.

→ child class cannot inherit the constructor of its parent.

→ super Refrence

→ Super keyword can be used to access some methods and variables that are not by defoolt inheriled by the child class.

→ Example constructor of the parent class and methods.

→ it is not possible to instinciale a parent class fom a child class directly but the soper keoyword can be used to do so.

Example!

→ Parent class

```
Public class School {
    Protected Str location;
    Public School (Str location) {
        this.location = location;
    }
    Pobic String toString() {
        return location;
    }
```

```
Public class classroom extends school {
    Private nom;
    Public classroom (int nom) {, Str add)
        Super(add);
        this.nom = nom;
    }
    Pobic String toString() {
        if (num > 0) {
            Sout ("Hello");
        } else { Super.toString(); }
    }
```

initializes Parent constructor

→ Java does not support multiple inheritance which means that a child class can only extend from one parent class.

→ overriding Methods

→ a child class can override a pre existing method that a parent class has meaning it can change a certain methods body. However the child class has to have the same exact header for the meathod as its parent.

→ final keyword can be used to prevent child from overriding a Parents method.

→ when the concept of overriding is applied to variables its called shadowing.

→ The Object class is default Parent of all the classes that exist in Java.

→ Abstract classes

→ you cannot create instance of Abstract classes.

→ To make an abstract the header should look as follows.

Pobiic abstract class className {
// ___ ___ ___

→ Abstract classes can have methods with no body.

→ ~~if atleast one of the method are method class~~

→ if a method has no body in an abstract class. the method should have an abstract keyword in its header.

→ However abstract classes can also have regular methods.

→ Abstract method cannot be declared final or static.

→ the child of an Abstract class should override the abstract methods of the parent else it will also ~~become~~ be considered abstract.

→ Interface won't give error

→ in case of an interface, all the methods and variables have to be abstract meaning they do not have a body.

→ the child class that implements a interface has to have rewrite methods body.

→ methods in object class
toString()
equals()
clone()

→ How to use interface

```
public interface Animal {
    // methods with no body
}
```

→ child class that impliments the interface

```
public class Example implements Animal {
    // all the methods need to be given
    a body
}
```

→ if the child class does not give body to the methods in the interface it has implemented from java throws an error.

→ if there are variables inside an interface they have to be final or they have to have a value even final variables need to have a value.

→ nothing inside a interface can be protected.

→ in case of an abstract class things can be protected.

→ if a method does not have a body it should be declared abstract.

→ it can have variables with no values.

→ if the child calass does not give body to the abstract methods of the abstract class it is not able to be instantiated. it is consided abstract (no error)

~~Polymorphism via interface~~

→ Polymorphism via inheritance

→ an object refrence ~~can refer to~~ an object of any class as long as it is related to it by inheritance.

Example:

if we have a parent class named "Parent" and a child class named "child"

The object of the child class can have a parent refrence

```
Parent x = new child();
```

Polymorphism via interface

```
public class Main {
    public static void main(String[] args){
        TestChild child1 = new TestChild();
        TestParent parent1 = new TestParent();

        System.out.println(child1.parentsMethod());
        System.out.println(child1.childMethod());

        System.out.println(parent1.parentsMethod());
        // System.out.println(parent1.child1Method()); this gives error because parent class does not have access to
child methods

        TestParent parent2 = new TestChild();

        System.out.println(parent2.sameMethod());
        System.out.println(parent1.sameMethod());

        // System.out.println(parent2.childMethod()); // This gives error because the compiler checkes of childMethod
in the parent class wich is not so it gives error

        System.out.println(((TestChild)parent2).childMethod()); // This works because now we hhave casted paarent to
TestChild type
    }
}
```

This is the parent method

This is a child method

This is the parent method

This is the same method from the child class

This is the same method from the parent class

This is a child method

→ a class can impliment multiple interfares.

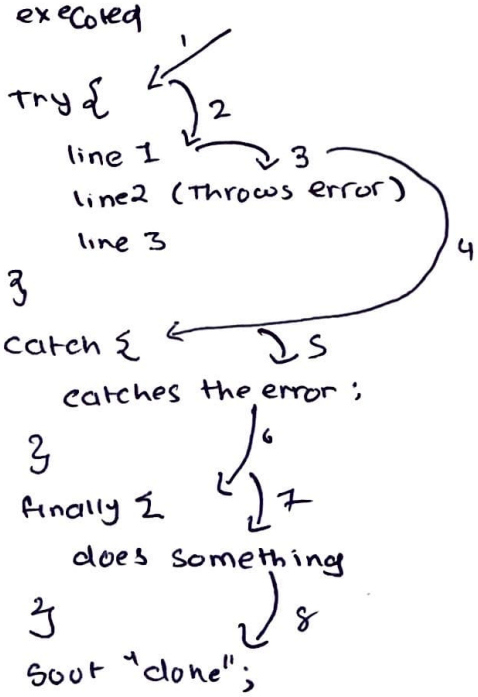
→ Exception Handling

→ try catch statements

```
try {
  //code
}
catch( Exception e){
  // code
}
finally {
   //code
}
```

→ The Finally claose is always execuled regardless of wether an error is occured or not.

→ How the try catch finally block is execoled



If no error is thrown in the try block everything in ran

Import Exceptions

→ StringIndexOutOfBoundException (unchecked)
occurs when we try to locale a char inside a string using .charAt(Index) and that index DNE

→ NumberFormatException (unchecked)
occurs when we use things like ~~ParseInt~~ Integer.ParseInt()
and the string cannot be converted to a nomber

→ Arethmetic Exception
Diversion by zero

→ All the exception classes are decendents of the throwable class

→ the only unchecked exception in Java are objects of type RontTme Exception or any of its decendents.

→ NullPointerException → (unchecked)

→ NoSuchMethodException → (checked)

→ classNotFoundException (checked)

→ throwing an Exception

we can create our own Exception in java

```
Pubic class ourException extends Exception {
        ourException (string e) {
            super (e);
    }   }
```

this class can be used as follows

```
Pubic class trialClass {

    Pubic static void main(String[] args) throws ourException {
        our Exception error = new ourException ("our error");
        int i = Scanner.nextInt();
        if (i = 7) {
            throw error}
        }
        sout "Hello" →
    }
}
```

this might never be executed if error is thrown

if you are trying to throw an error out side try statement yo need to have throws (Exception class name) in the method header that throws exception example main method.

```
public static void main (String[] args). throws exceptionclass name {

    }
```