Scanner(can work in sts and on terminal), Console(works only for terminal)

```
Scanner sc = new Scanner(System.in);
sc.next();

Console console = System.console();
console.readLine();
```

Camel Case
Pascal Case

Own Rules
Own Syntax

Datatypes
Primitive (Value type)
- Boolean
  - boolean
- Character
  - char
- Integral
  - byte, short, int, long
- Floating-Point
  - float, double

Non-Primitive (Reference type)
- Array
- Class
- Enum
- Interface

Wrapper Class

byte -> convert -> short -> Widening
short -> convert -> int -> Widening
int -> convert -> long -> Widening

long -> convert -> int -> Narrowing
int -> convert -> short -> Narrowing
short -> convert -> byte -> Narrowing

long -> convert ->float -> widening
float -> convert -> long -> narrowing

byte -> convert -> char -> Type Conversion
char -> convert -> byte -> Type Conversion

int ->convert -> Integer -> Boxing

Integer -> convert -> int -> UnBoxing

```
int n1 = 10;

Integer i1 = new Integer(n1); // Boxing
Integer i2 = n1; // Auto-Boxing

Integer i3 = new Integer(20);
int n2 = i3.intValue(); // UnBoxing;
n2 = i3; // Auto-UnBoxing
```

```
class Date{

}

class Time{

}

class Date{

}
```

```
namespace na{
int num1 = 10;
}

namespace nb{
int num1= 100;
}
```

pacakge
- It is a container which is used to
1. Organise the code
2. Resolve the ambugity

Date -> Utility class

```
package utils{

class Date{
};

}
```

Date -> sql

```
package sql{

class Date{

};

}
```

1. Create a directory cmd_line
2. create a project(directoy) called as demo01
3. create 2 directories inside demo01 as src and bin
4. Inside src add a Program.java file with package declaration

give the below command from src directory

```
javac -d ../bin Program.java
```

to Execute set the classpath

```
export CLASSPATH=../bin
```

to run

```
java p1.Program
```

```
demo01
  |
  |__src
  |    |__Program.java (package p1;)
  |
  |__bin
       |__p1
           |__Program.class
```

```
demo02
  └─ src
      ├─ Program.java -> pacakage p1;
      └─ Time.java -> package p2;
     bin
      ├─ p1
      │   └─ Program.class
      └─ p2
          └─ Time.class
```

1. compile the Time.java
javac -d ../bin Time.java

2. set the classpath
export CLASSPATH=../bin

3. import the Time class in Program.java file
in program.java file write the below statement
import p2.Time;

4. compile the Program.java
javac -d ../bin Program.java

5. Execute
java p1.Program

sunbeam.com

com.sunbeam

cloud server

com.sunbeam.
attendance

com.sunbeam.
latecommers

Google Playstore

com.sunbeam.attendance

com.sunbeam.latecommers

com.sunbeam.lateattendance

com.zomato.foodorder

com.swiggy.foodorder

com.cdac.attendance
com.cdac.latecommers

package
comapny domain name in reverese order . projectname . component

com.sunbeam.attendancesystem.entity
com.sunbeam.attendancesystem.tester

calculate area for different shapes
2d (circle, rectangle), 3d(Box)

sunbeam

com.sunbeam.shapes.2d
com.sunbeam.shapes.3d

package <packagename>;

companydomain name in reverese order.projectname.component

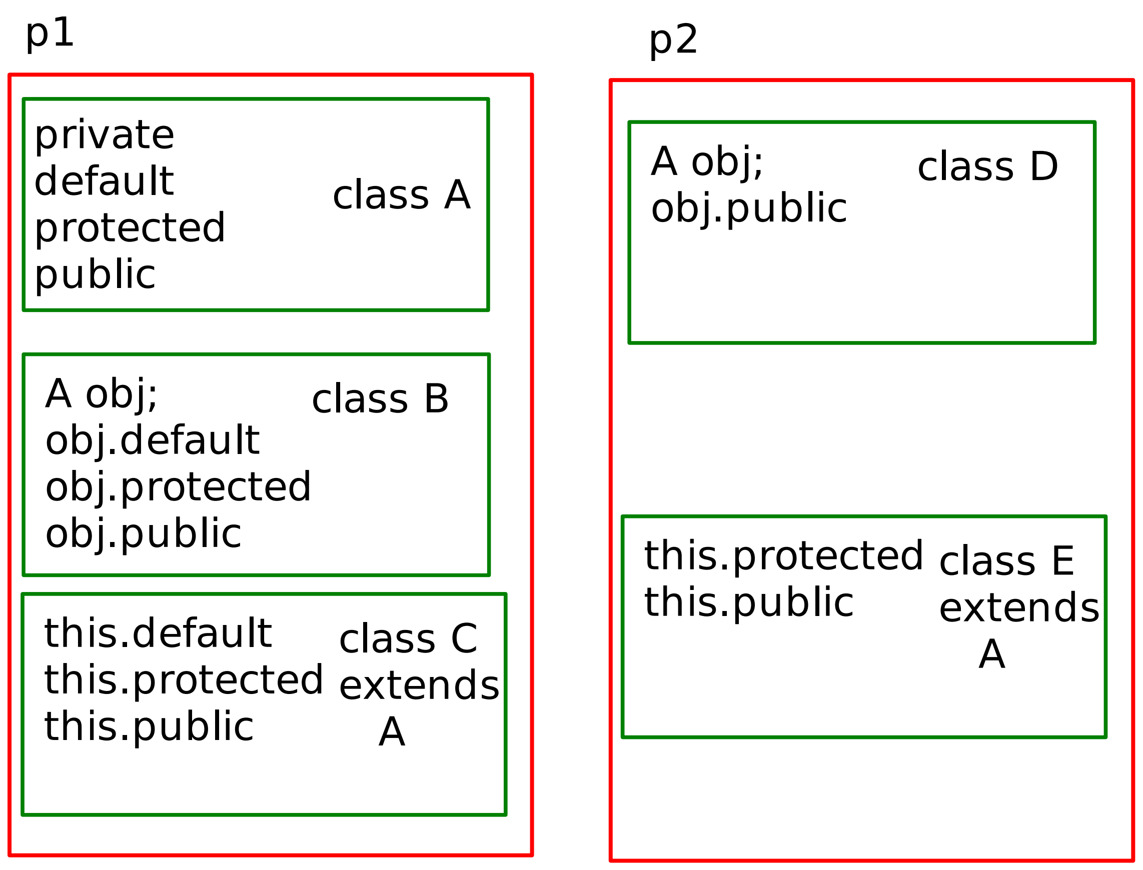
4:15 -> pushed
6:45 -> push
10Pm -> push

How to create a package

AccessModifiers
private
protected
default
public

package p1;
class Test{
private
protected
public
default
}

Access Specifiers
private
protected
public



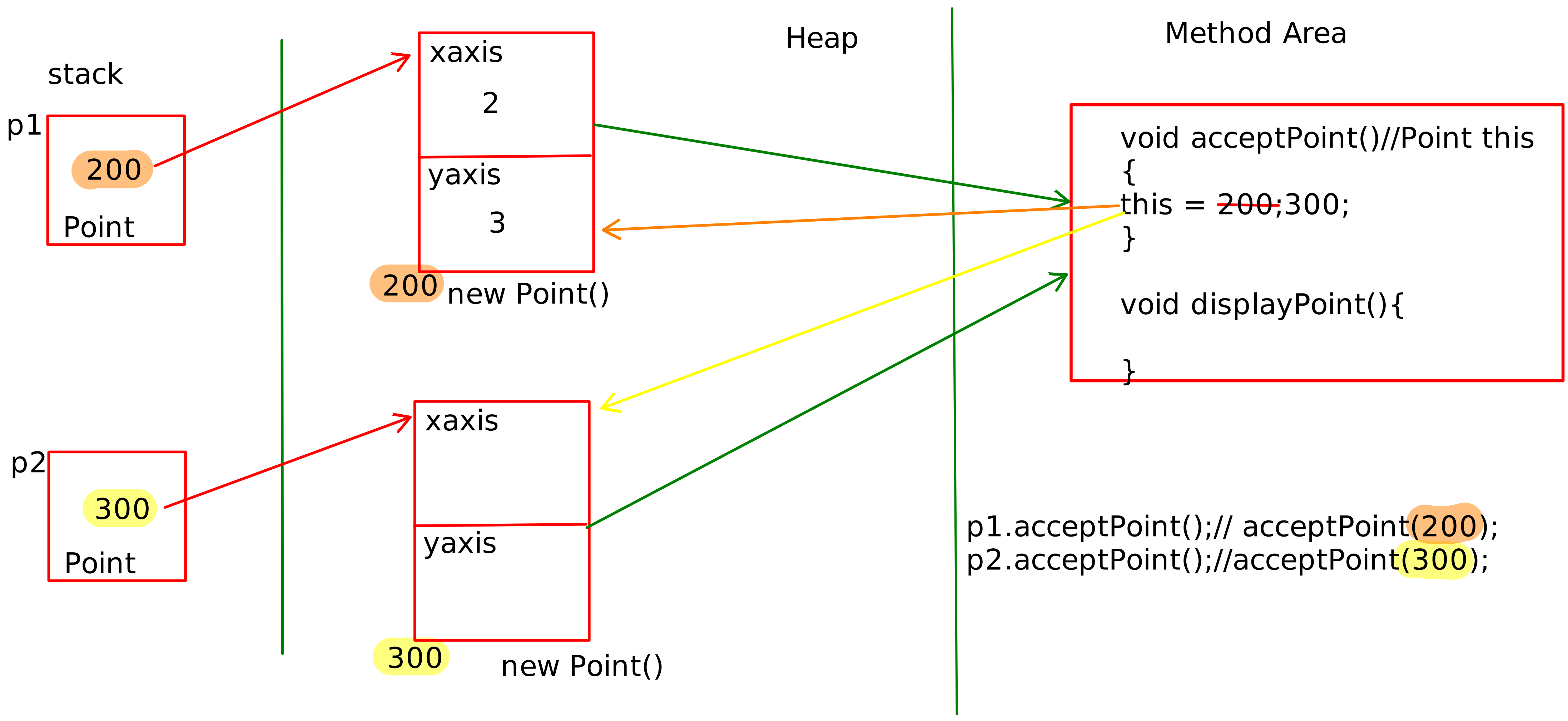
Same Package (P1)

Within the class-> All
Other class ->
- Except Private All members on class object
Subclass ->
- Except Private All members directly

Different Package(p2)
Other class ->
- Only public are accessiable using class object

subclass ->
- only protected and public are accessiable directly

Public class
- As per java langauage specification
1. public class must be defined in its own file. i.e name of public class and .java file name should be same
2. public classes are used to provide their visibility acrosss multiple different packages



```
void acceptPoint()//Point this
{
this = ~~200~~;300;
}

void displayPoint(){

}
```

p1.acceptPoint();// acceptPoint(200);
p2.acceptPoint();//acceptPoint(300);

Point p1 = new Point();
Point p2 = new point();

Types of Methods
1. Constructor
2. Setters
3. Getters
4. Facilitators

Constructor
- It is a special method of a class
- why is is special ?
    - Its name is same as that of classname
    - it do not have any return type
    - It gets automatically called when object is created

- types of ctor
1. Default/Parameterless
2. Parameterized ctor

Setters ->
- To provide write permission for the individual private field of the class
- Setter method should always start with the name set follwed by the name of field.
- Setters should not return anything.
- Setters should accept 1 parameter of the same type as that of the field to write.

Getters ->
- To provide read permission for the individual private field of the class
- Getter method should always start with the name get follwed by the name of field.
- Getters should return value of that individual field.
- Getters should not accept any parameter

Facilitator ->
- Any method that is used to perform the operations on all or some fields of the class are called as Facilitators
- Facilitators are used to write the business logic

Constructor Chaning ->
- To call the existing constructor in the class from another constructor is called as constructor chaning
- Constructor chaning can be done using `this statement` [this()].
- this statement should be the first statement in the constructor body.