Java access modifiers.
Classes, fields, constructors and methods can have one of four different Java access modifiers:

- private
- default (package)
- protected
- public

private Access Modifier

If a method or variable is marked as `private` (has the `private` access modifier assigned to it), then only code inside the same class can access the variable, or call the method. Code inside subclasses cannot access the variable or method, nor can code from any external class.

Classes cannot be marked with the `private` access modifier. Marking a class with the `private` access modifier would mean that no other class could access it, which means that you could not really use the class at all.

```
public class Student{

  String name;

  double cgpa;

}
```

```
public class Student{

  public String name;

  public double cgpa;

}
```

```
public class StudentTester{

  public static void main(String[] args){

    Student mahrin = new Student();

    System.out.println(mahrin.name);

    System.out.println(mahrin.cgpa);
  }

}
```

By default all the properties have public access, so writing public in front of it does not make any difference and it can be accessed from any class.

When we make the instance variables private, we cannot access it from the Tester class.

ERROR: name has **private access in Student**

```
public class Student{

  private String name;

  private double cgpa;

}
```

So only code inside the same class can access the variable, or call the method. SO written PleasePutTheNameInStudent(String n), PleaseGiveTheNameInStudent().

```
public class Student{

  private String name;

  private double cgpa;



  public void PleasePutTheNameInStudent(String n){

    name=n;

  }

  public String PleaseGiveTheNameInStudent(){

    return name;

  }
}
public class StudentTester{
  public static void main(String[] args){
    Student mahrin = new Student();
    mahrin. PleasePutTheNameInStudent("mahrin");
    System.out.println(mahrin.PleaseGiveTheNameInStudent());

//    System.out.println(mahrin.name);
//    System.out.println(mahrin.cgpa);
  }
}

/////////////////////////////////////////////////
public class Student{
  private String name;
  private double cgpa;

  public void setName(String n){
    name=n;
  }
  public String getName(){
    return name;
  }

}
```

```
public class StudentTester{

  public static void main(String[] args){

    Student mahrin = new Student();

    mahrin.setName("mahrin");

    System.out.println(mahrin.getName());

  }

}
```

two methods `getName()` and `setName(String n)` can access the `name` variable. The two methods are declared `public`, meaning they can be called from code anywhere in your application.

private Constructors

If a constructor in a class is assigned the `private` Java access modifier, that means that the constructor cannot be called from anywhere outside the class. A `private` constructor can still get called from other constructors, or from `static` methods in the same class

```
public class Student{

  private String name;

  private double cgpa;

  private String id;

  private Student(){

    name="Default Student";

    cgpa=0.0;

  }

  public Student(String id){

    this();

    this.id=id;

  }

  public void setName(String n){

    name=n;

  }
  public String getName(){
    return name;
  }
  public double getCgpa(){
    return cgpa;
  }
}

public class StudentTester{
  public static void main(String[] args){
    Student mahrin = new Student("1234");
    //mahrin.setName("mahrin");
    System.out.println(mahrin.getName());
    System.out.println(mahrin.getCgpa());
  }
}
```