Learning Outcome (LO)

STT	LO	SO/PI
1	Hiện thực các bài toán liên quan đến danh sách liên kết, cây	
2	Hiện thực các bài toán liên quan đến cây nhị phân	
3	Hiện thực các bài toán tìm kiếm tối ưu	
4	Hiện thực một số bài toán trong Trí tuệ nhân tạo	

REVIEW

1. Java is an Object Oriented Programming Language (98%)
2. Class and Object

WHY?

- Viết chương trình nhập dãy số thực, sắp xếp, tìm kiếm, thêm xoá
- Viết chương trình quản lý thông tin khách hàng

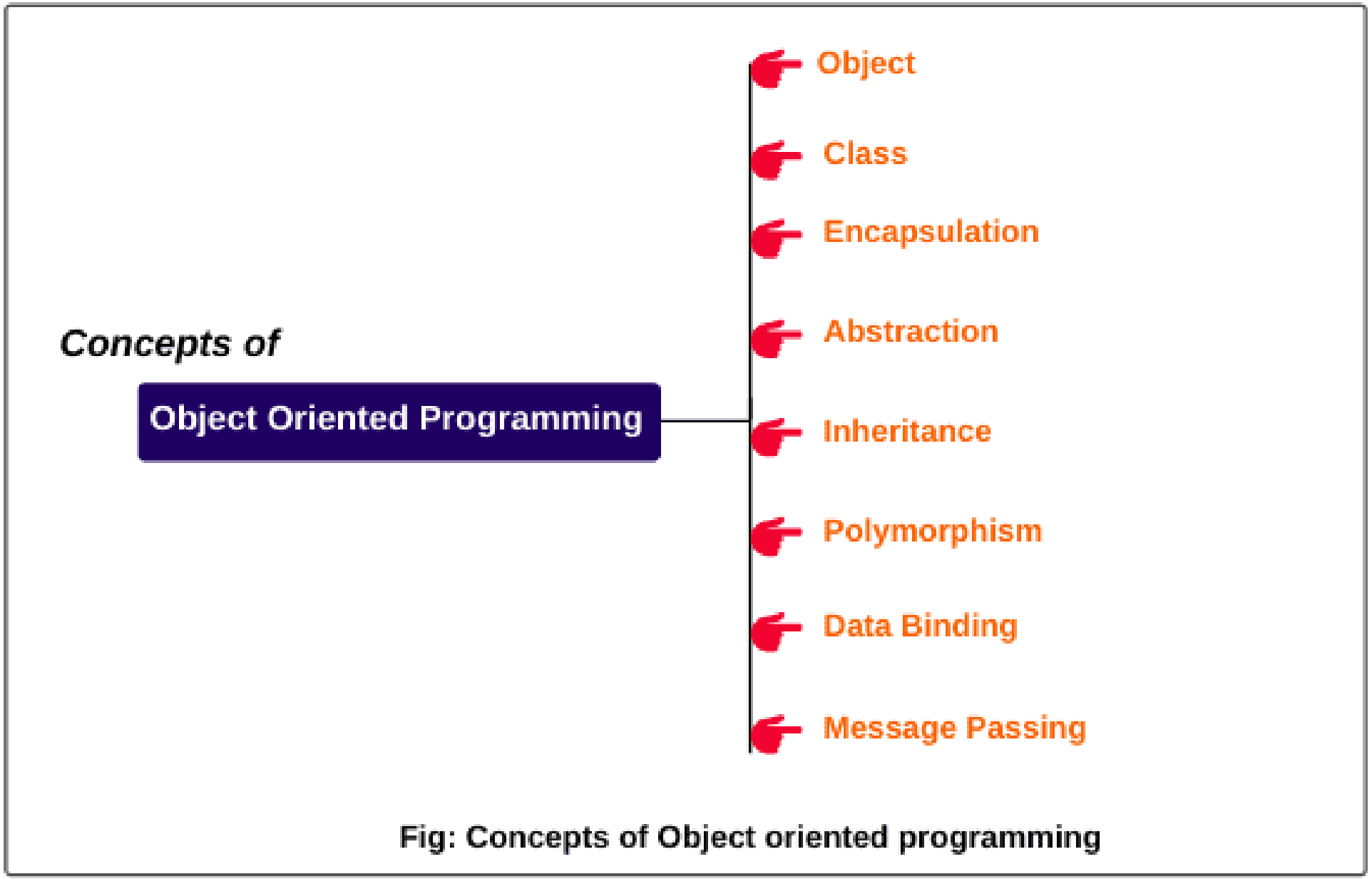
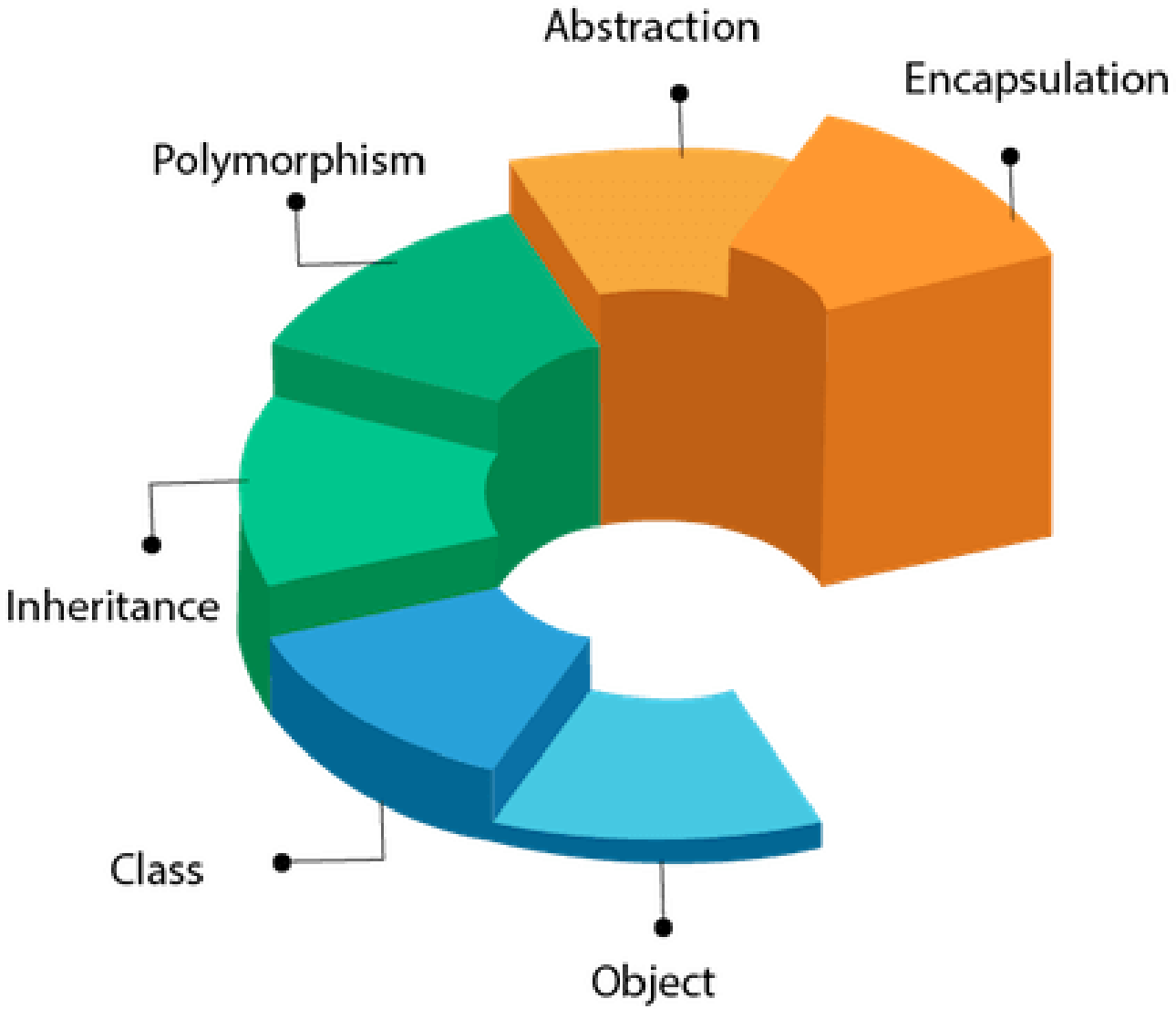


Fig: Concepts of Object oriented programming

OOPs (Object-Oriented Programming System)



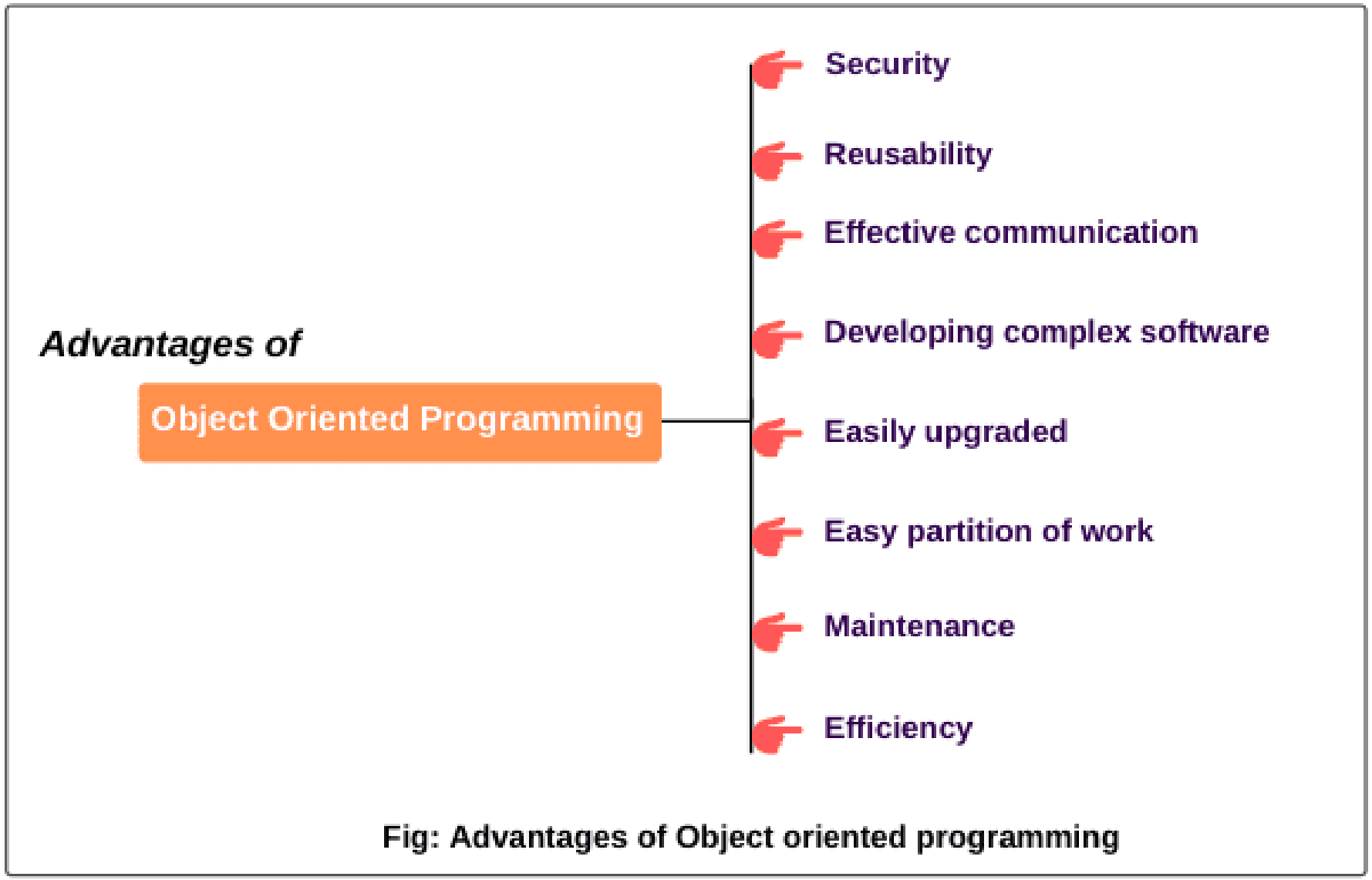
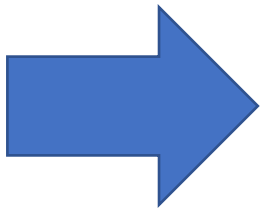


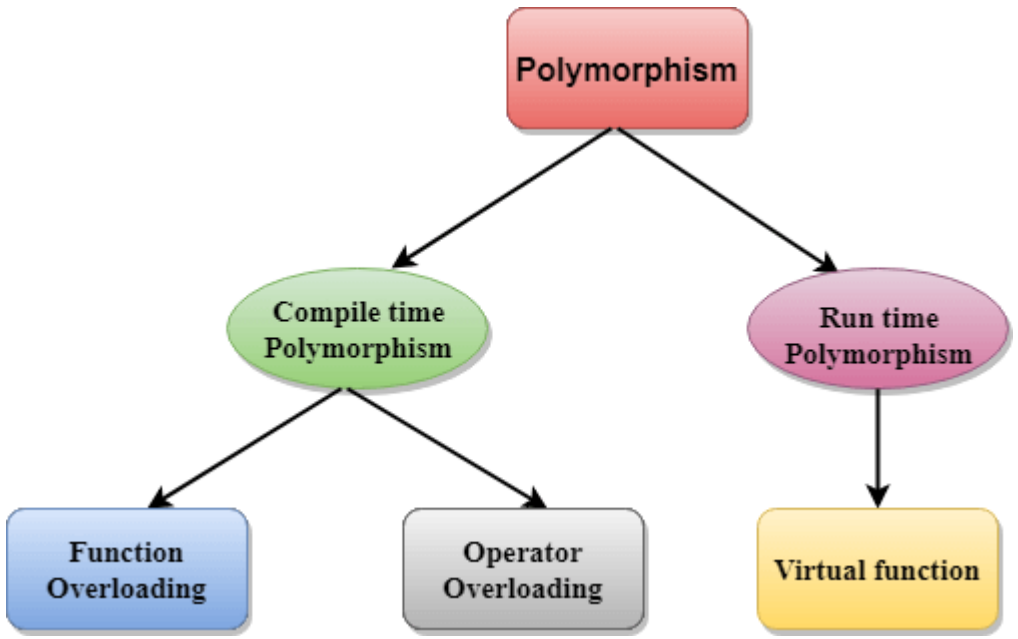
Fig: Advantages of Object oriented programming

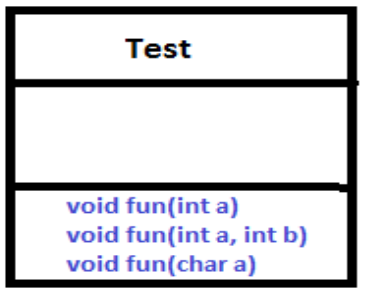
INHERITANCE

```
class A {
 void m1(){
  . . . .
 }
}
class B {
 void m1(){
  . . . .
 }
 void m2(){
  . . . .
 }
}
class C {
 void m1(){
  . . . .
 }
 void m2(){
  . . . .
 }
 void m3(){
  . . . .
 }
}
```

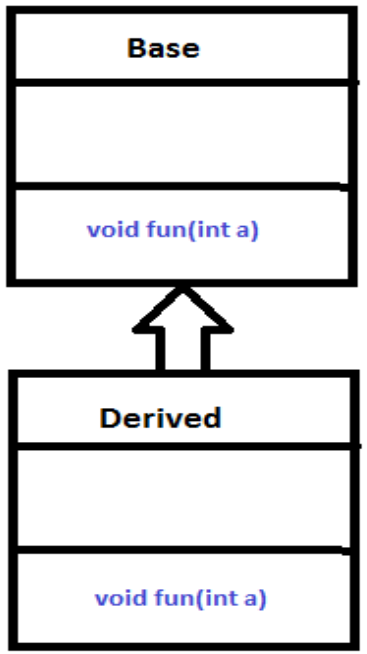


```
class A {
 void m1(){
  . . . . .
 }
}
class B extends A {
 void m2(){
  . . . . .
 }
}
class C extends B {
 void m3(){
  . . . .
 }
}
```





Overloading

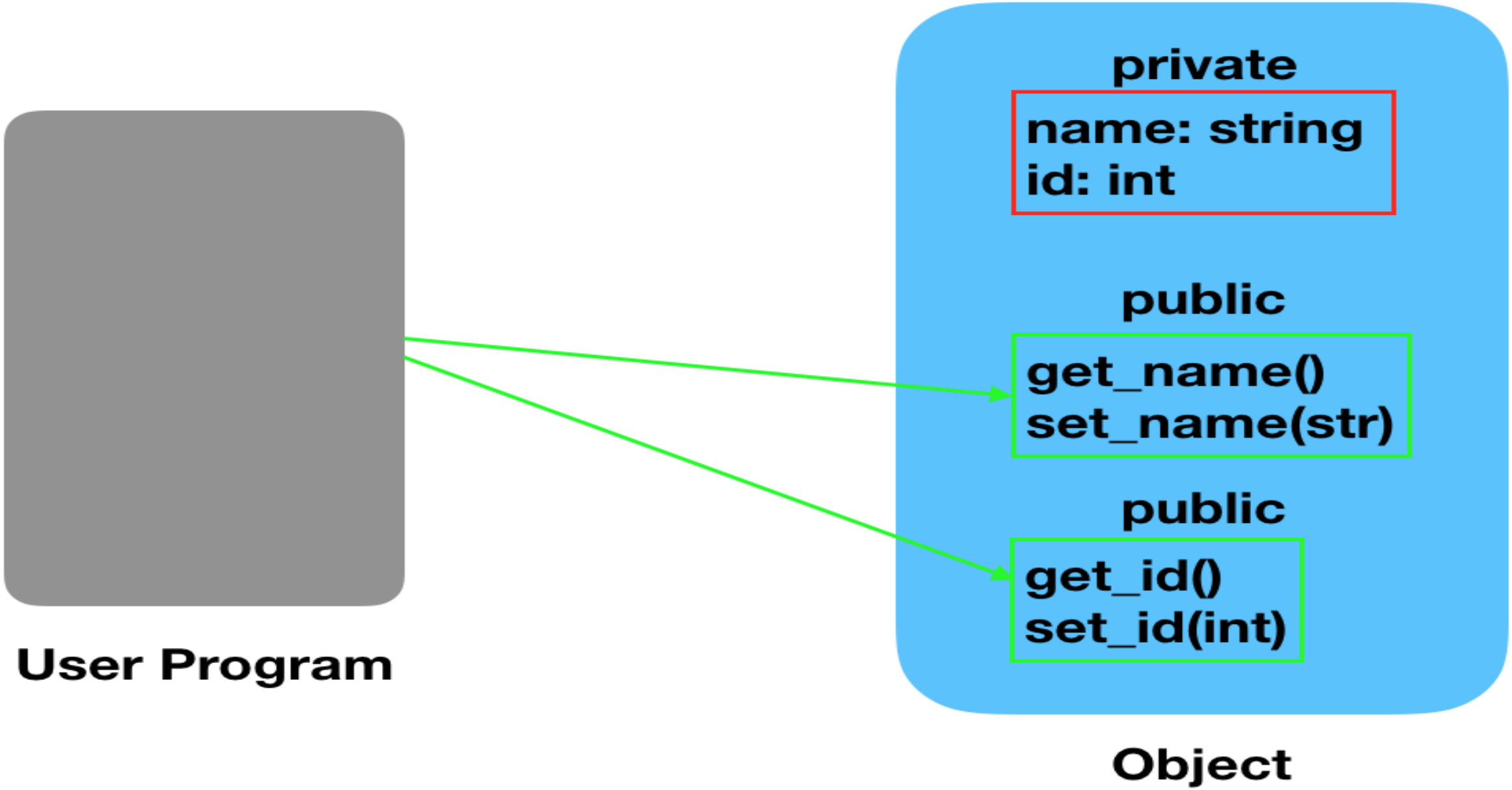


Overriding

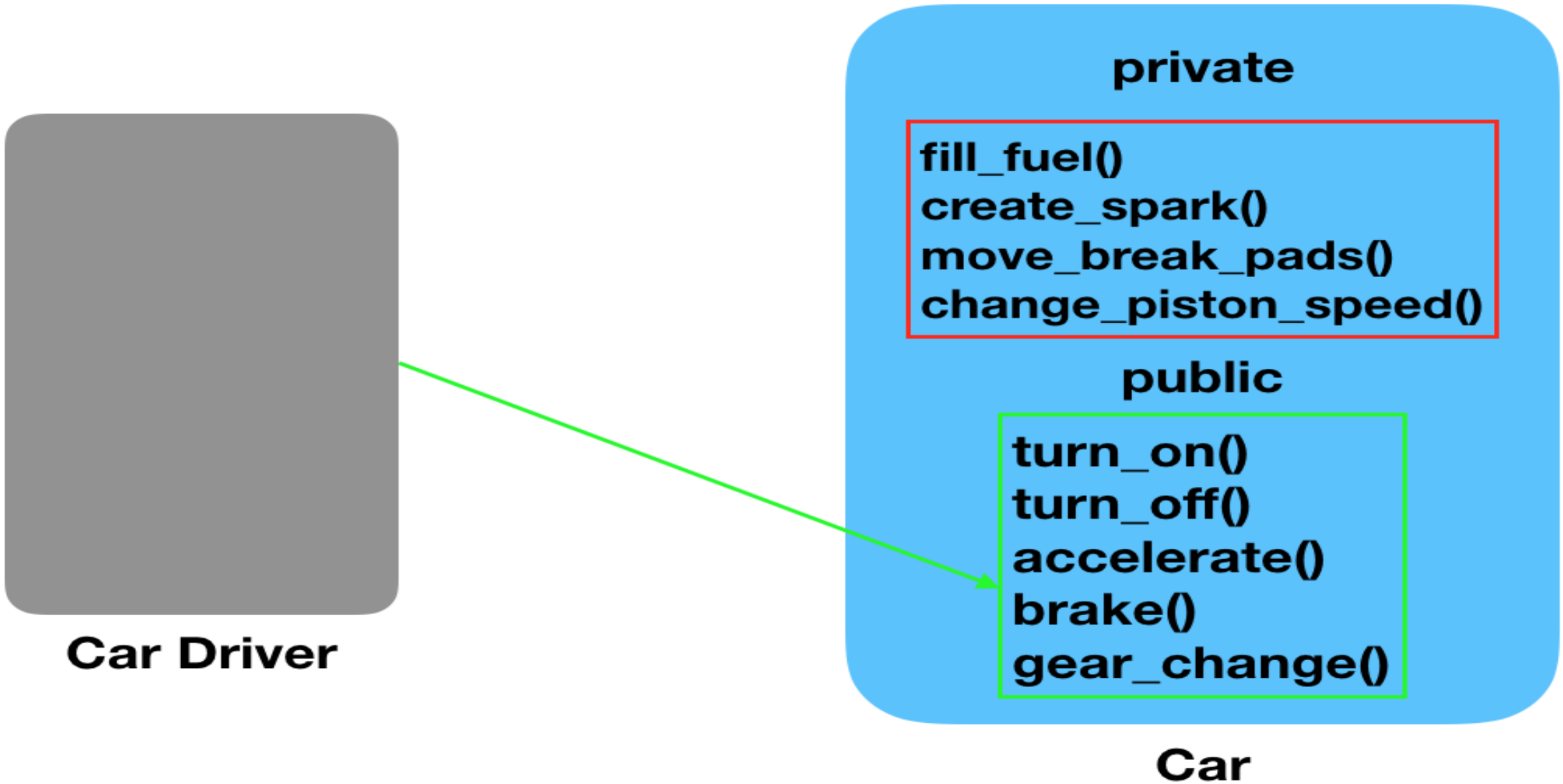
What is Abstraction in OOPS?

- Core Object-Oriented Programming Concept
- Process of hiding the internal data and implementation from the outer world.
- Two Types:
    - Data Abstraction
    - Process Abstraction

Data Abstraction



Process Abstraction



Abstration

- Quản lý phương tiện:
- Xe oto – abstraction(data) -> class oto
- Xe may – abstraction(data) -> class xemay
- Xe đạp – abstraction(data) ->class xedap
- Class oto,xemay,xedap -> abstraction class xeco

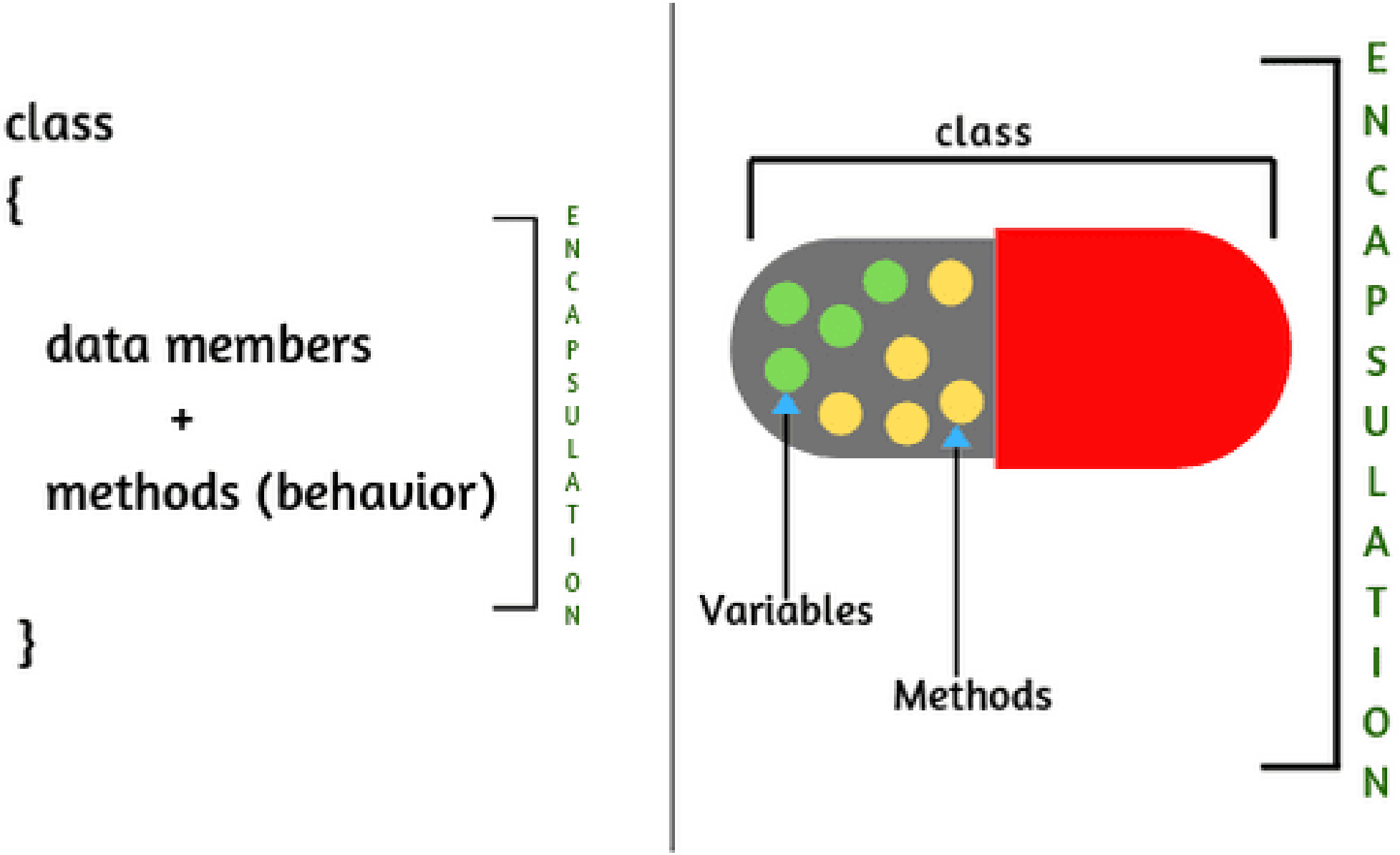


Fig: Encapsulation

Structure of a Java class

Package statement [Optional] [Must be the first line if written. Only execption is comments]

Import statement(s) [Optional]

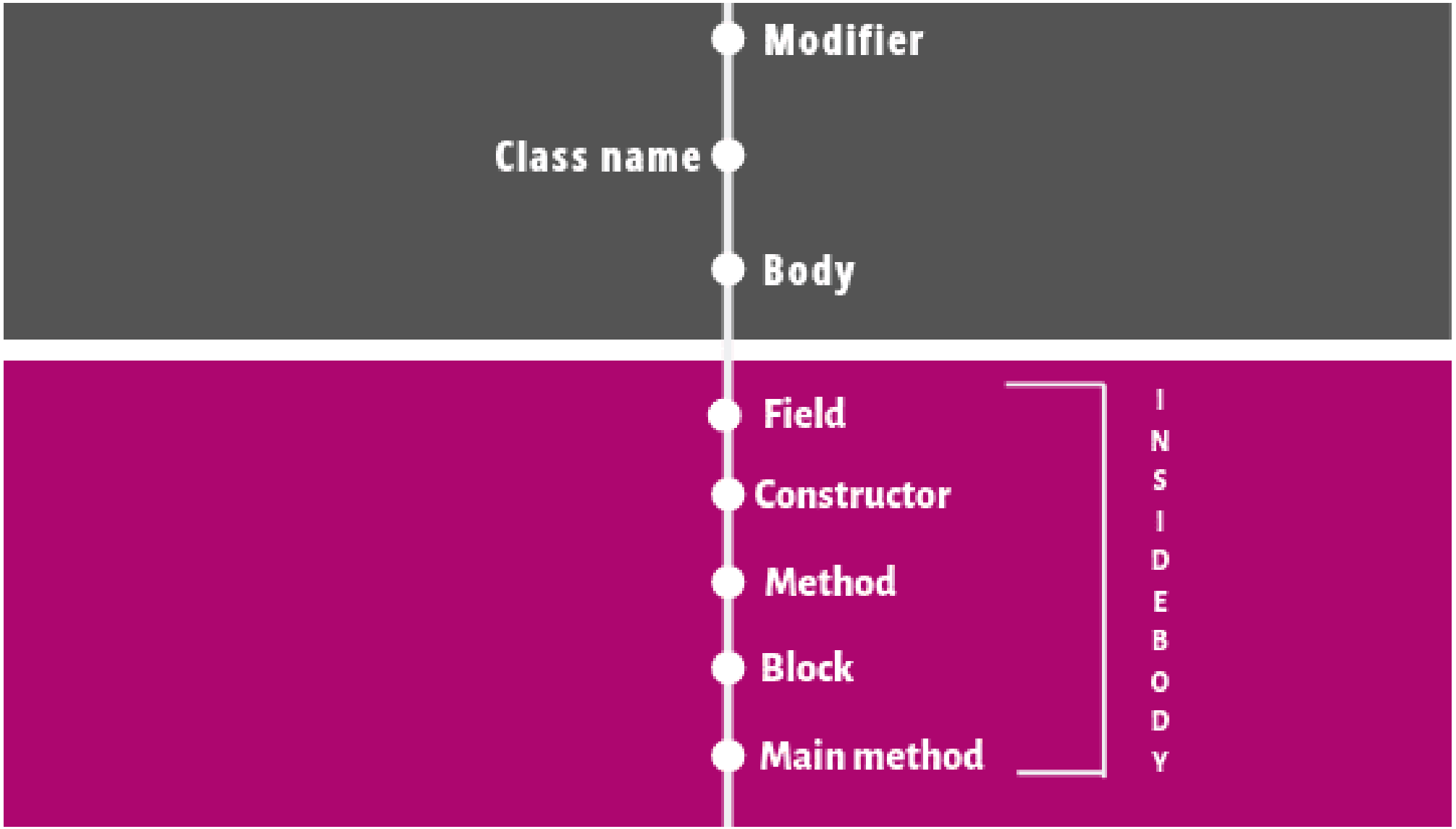
Comments [Optional] [Can be written anywhere in the code]

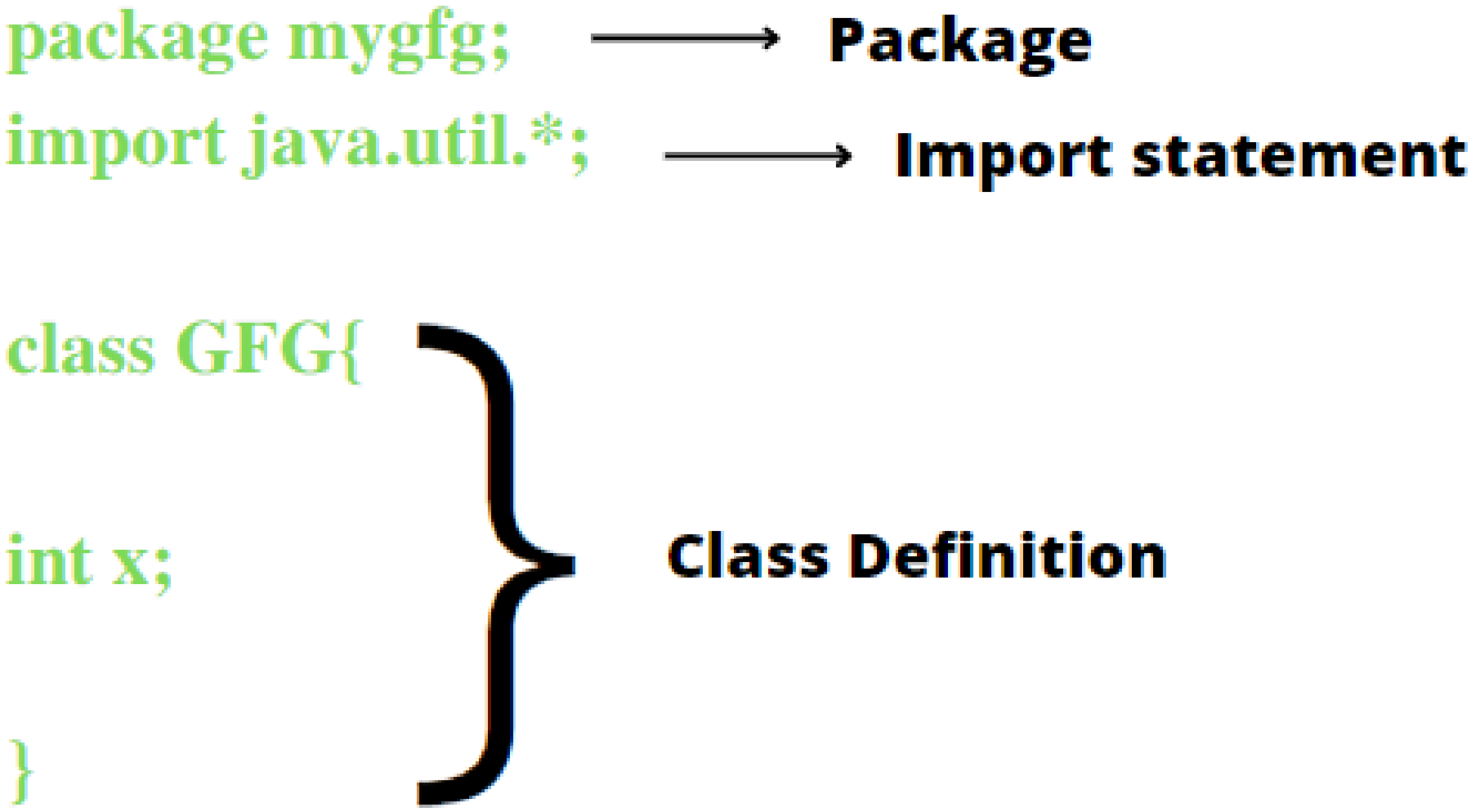
Class declaration
{
    Variable declarations
    Comments
    Constructors
    Methods
    Nested Classes
    Nested Interfaces
    Enums
}

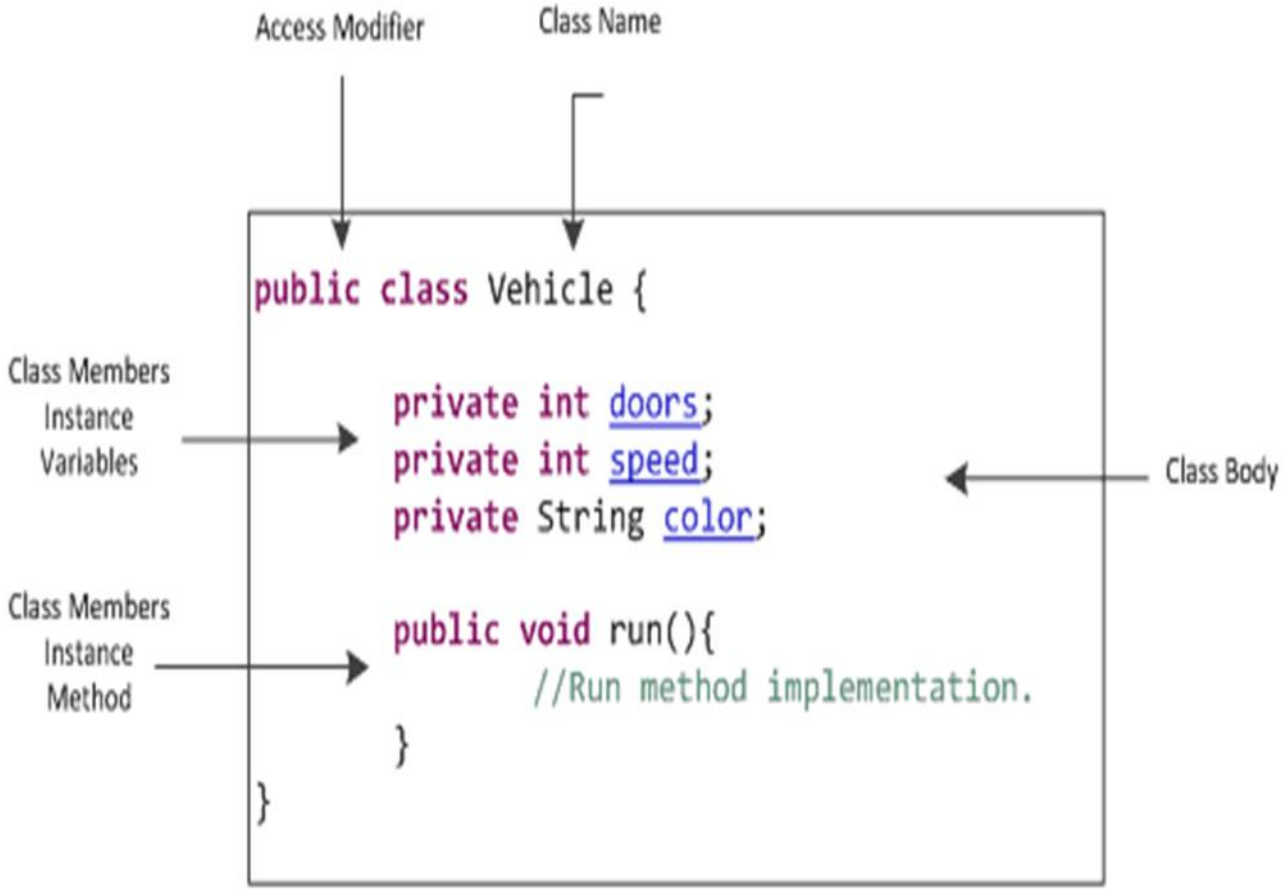
These things can be written in any order and all are optional

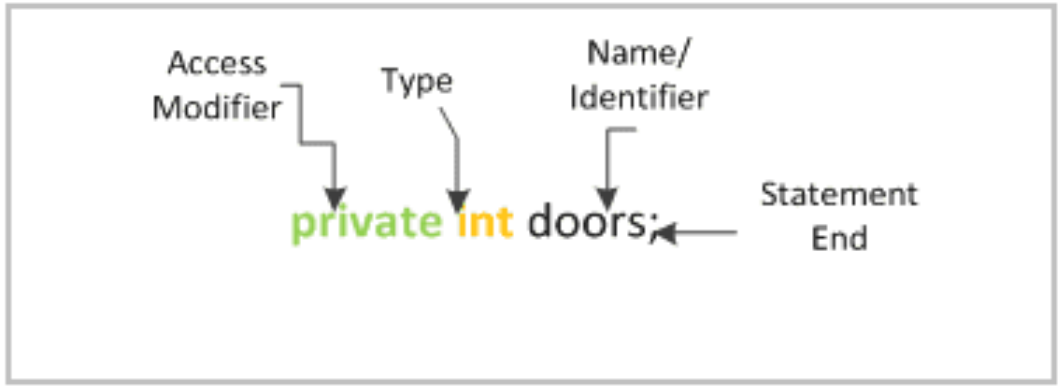
Class in Java

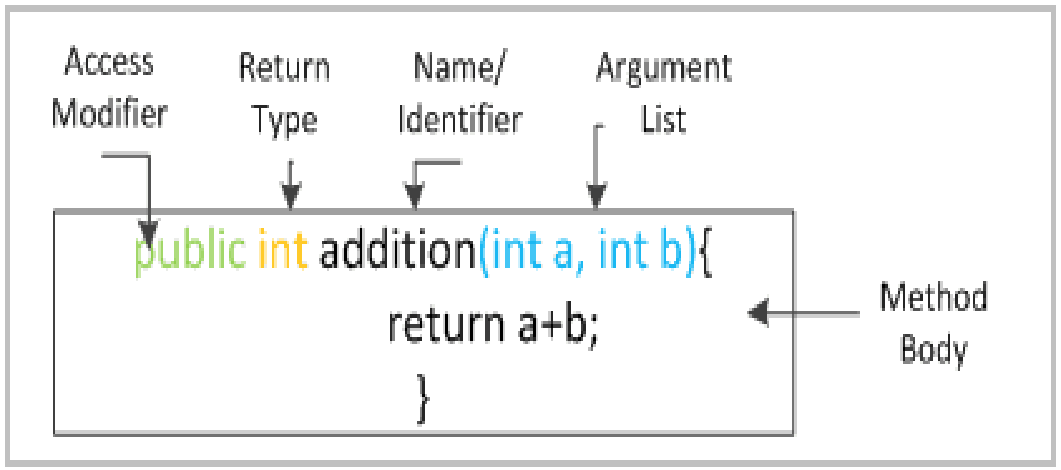
A class can have the following components to act as a template




package mygfg;
Package
import java.util.*;
Import statement
class GFG{
int x;
}
Class Definition

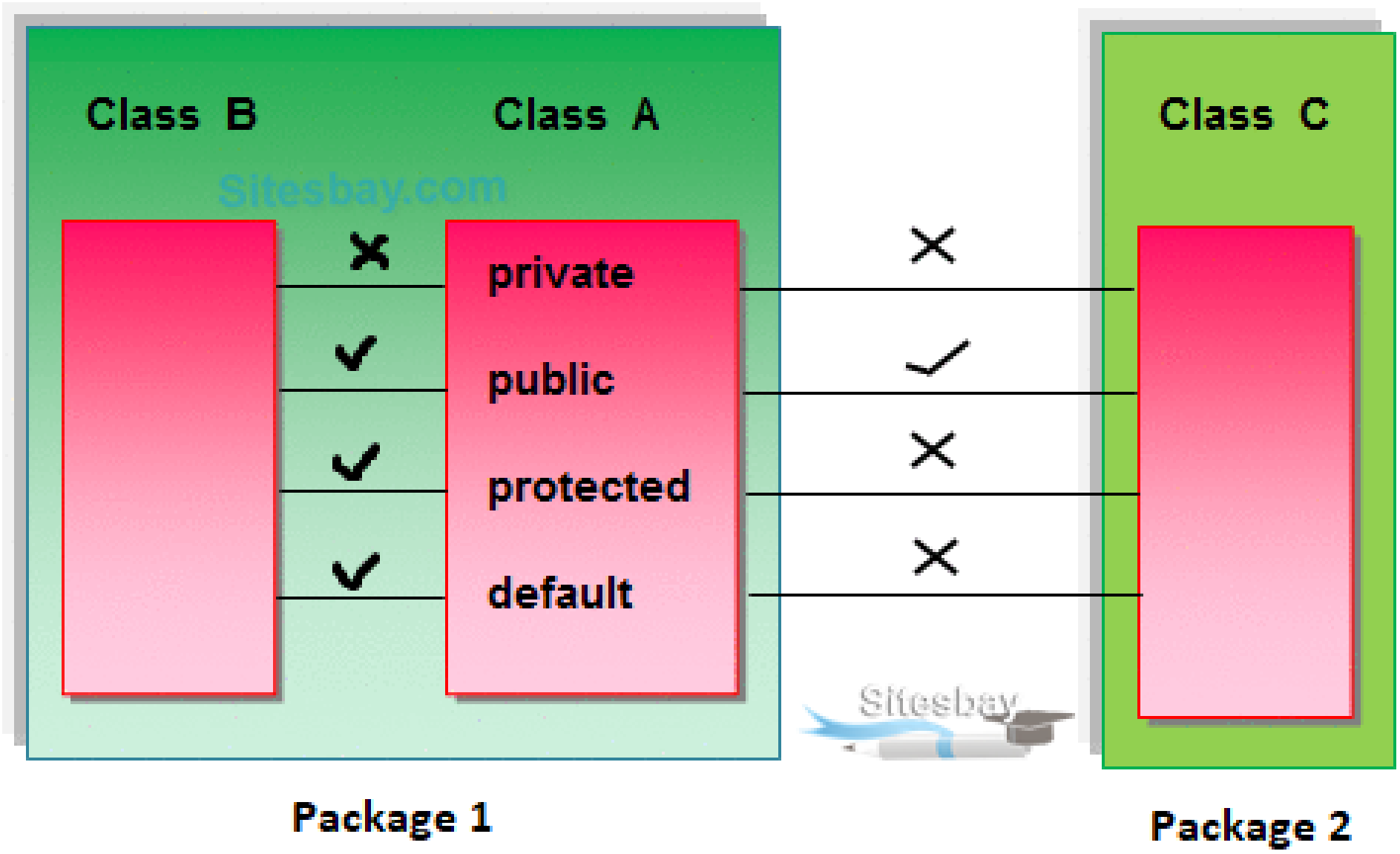

Access Modifier
Class Name
public class Vehicle {
Class Members Instance Variables
private int doors;
private int speed;
private String color;
Class Body
Class Members Instance Method
public void run(){
//Run method implementation.
}
}


Access Modifier
Type
Name/ Identifier
private int doors;
Statement End


Access Modifier
Return Type
Name/ Identifier
Argument List
public int addition(int a, int b){
return a+b;
}
Method Body

Access Modifier

Access Modifiers	Within the class	Within the Same Package	Subclass	In Other Packages
PUBLIC	Access Allowed	Access Allowed	Access Allowed	Access Allowed
PROTECTED	Access Allowed	Access Allowed	Access allowed	Access Denied
DEFAULT (NO ACCESS MODIFIER)	Access Allowed	Access Allowed	Access Denied	Access Denied
PRIVATE	Access Allowed	Access Denied	Access Denied	Access Denied


Class B
Class A
Sitesbay.com
private
public
protected
default
Class C
Sitesbay
Package 1
Package 2

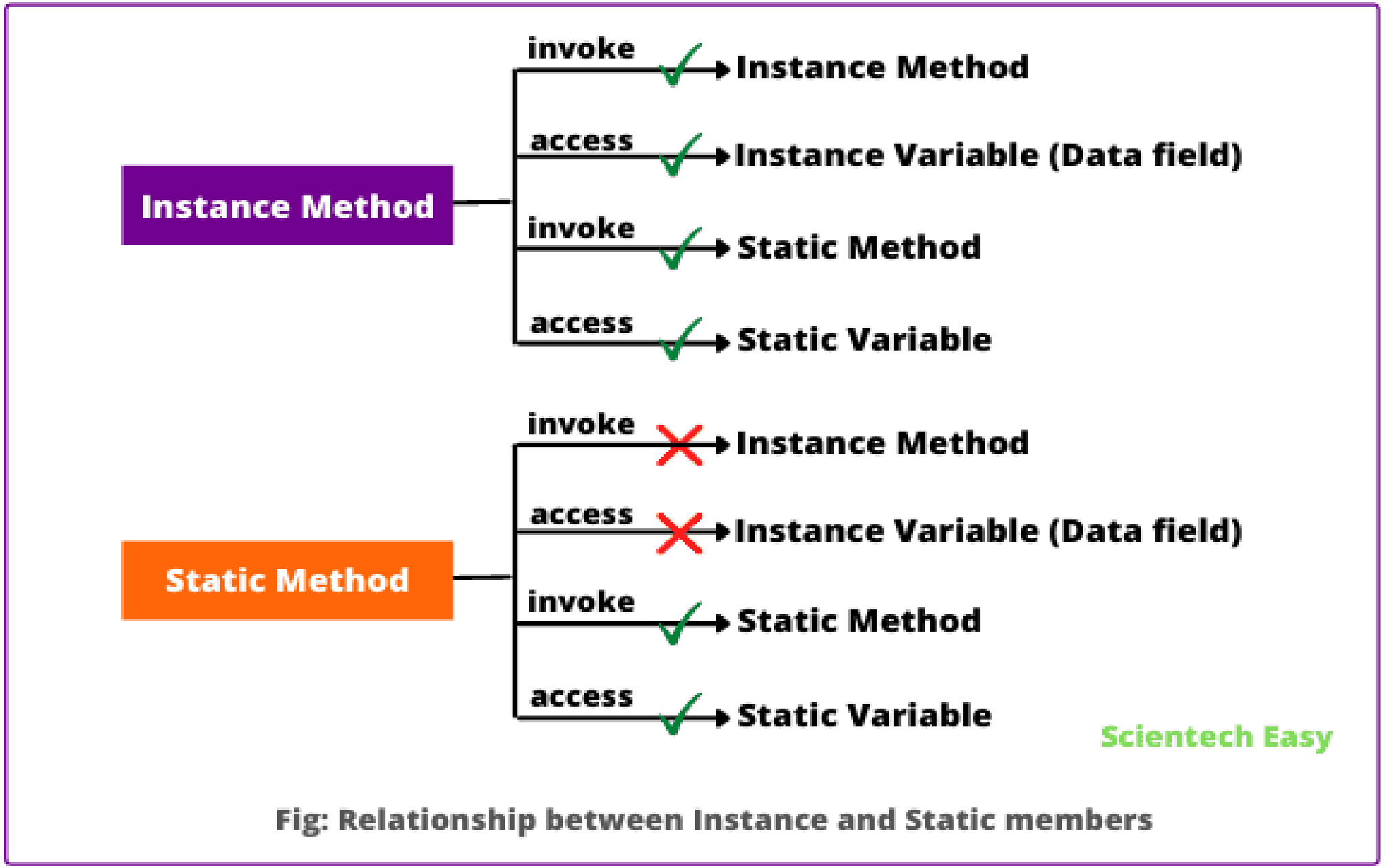


Fig: Relationship between Instance and Static members

Static method declaration in Java

1. public static void add()
{
- - - - - -
// Method body
- - - - - - -
}

2. public static void add(int x)
{
- - - - - -
// Method body
- - - - - - -
}

Modifiers

Return Type

3. public static int add()
{
- - - - - - - -
// Method body
- - - - - - - -
return statement
}

4. public static int add(int y)
{
- - - - - - - - - - -
// Method body
- - - - - - - - - - -
return statement
}

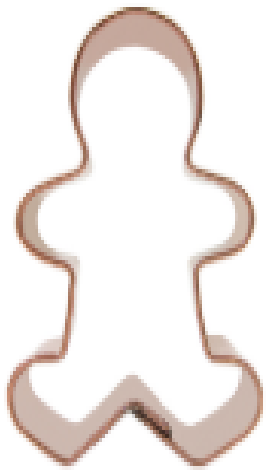
Local variable

```
public class Person
{
  /* instance variables for Person attributes */
  private String name;
  private String email;
  private String phoneNumber;

   /** a constructor to initialize the attributes
    for a Person object with the given parameters*/
  public Person(String initName, String initEmail,
                String initphoneNumber)
  { /* Implementation not shown*/ }

  /* method to print Person attributes */
  public void print()
  { /* Implementation not shown*/ }
}
```

Person Instance Variables

Person Constructors

Person Methods



Person Class

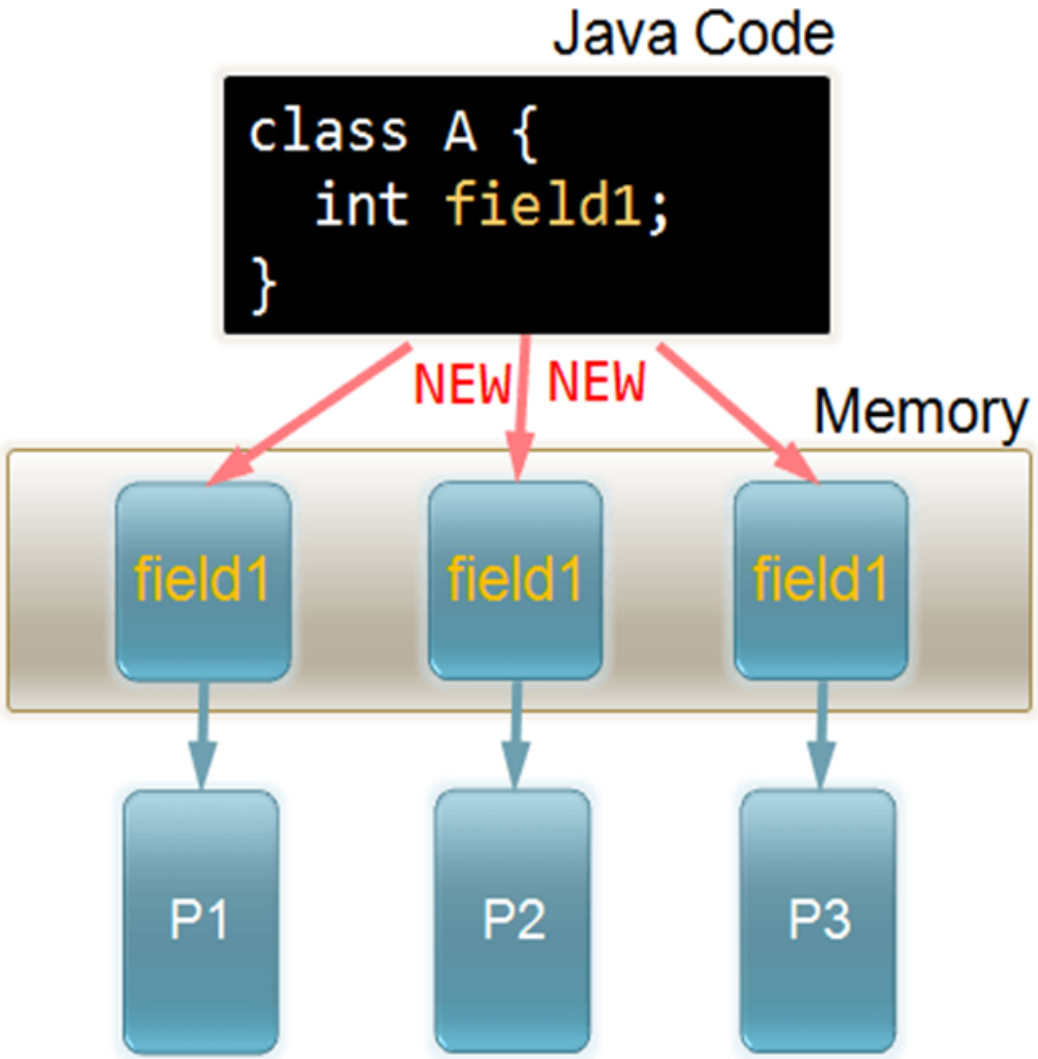
Private Encapsulated Data:
- -name
- -email
- -phoneNumber

Public Methods:
+Person(String n, String e, String p)
+print()

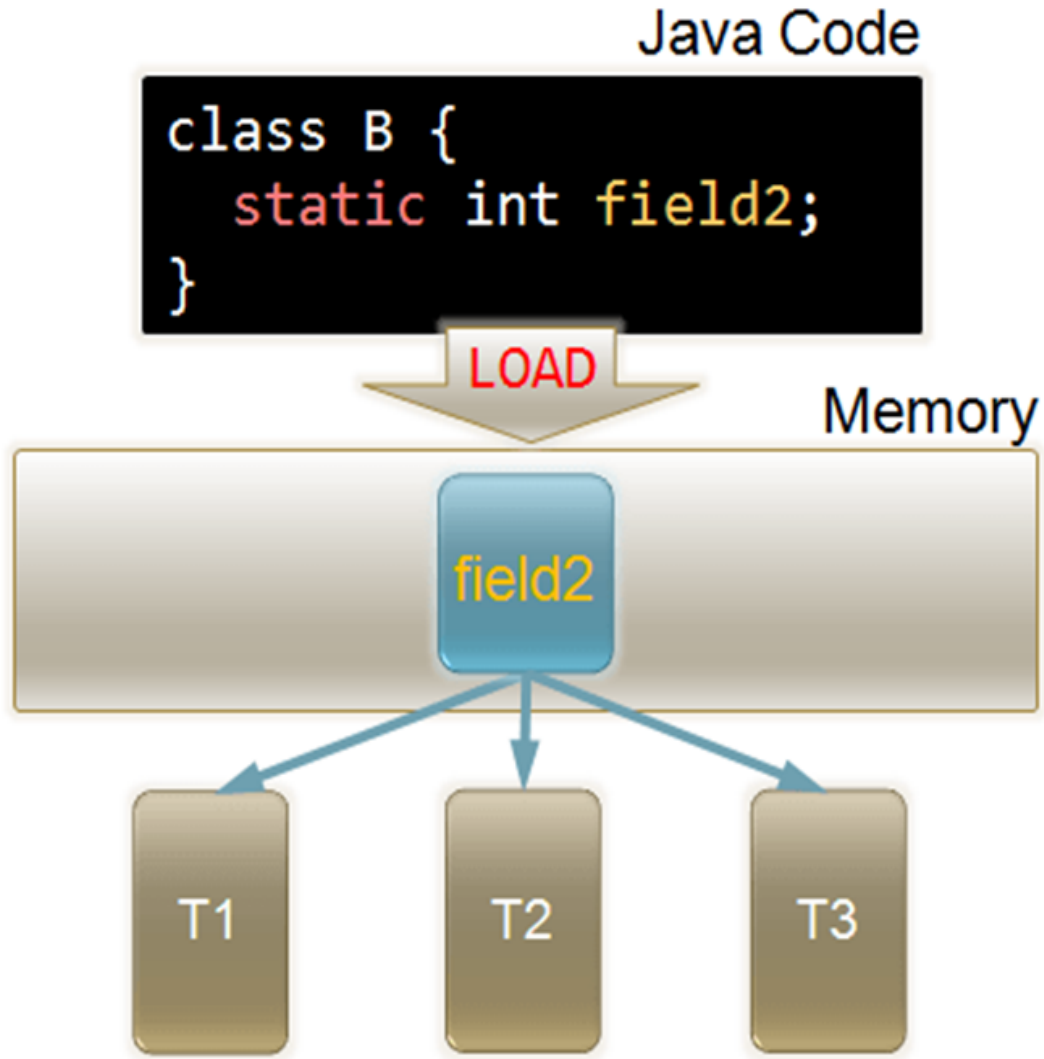
p1 object
---
name = "Sana"
email = "sana@gmail.com"
phoneNumber="123-456-7890"

p2 object
---
name = "Jean"
email = "jean@gmail.com"
phoneNumber="404- 899-9955"

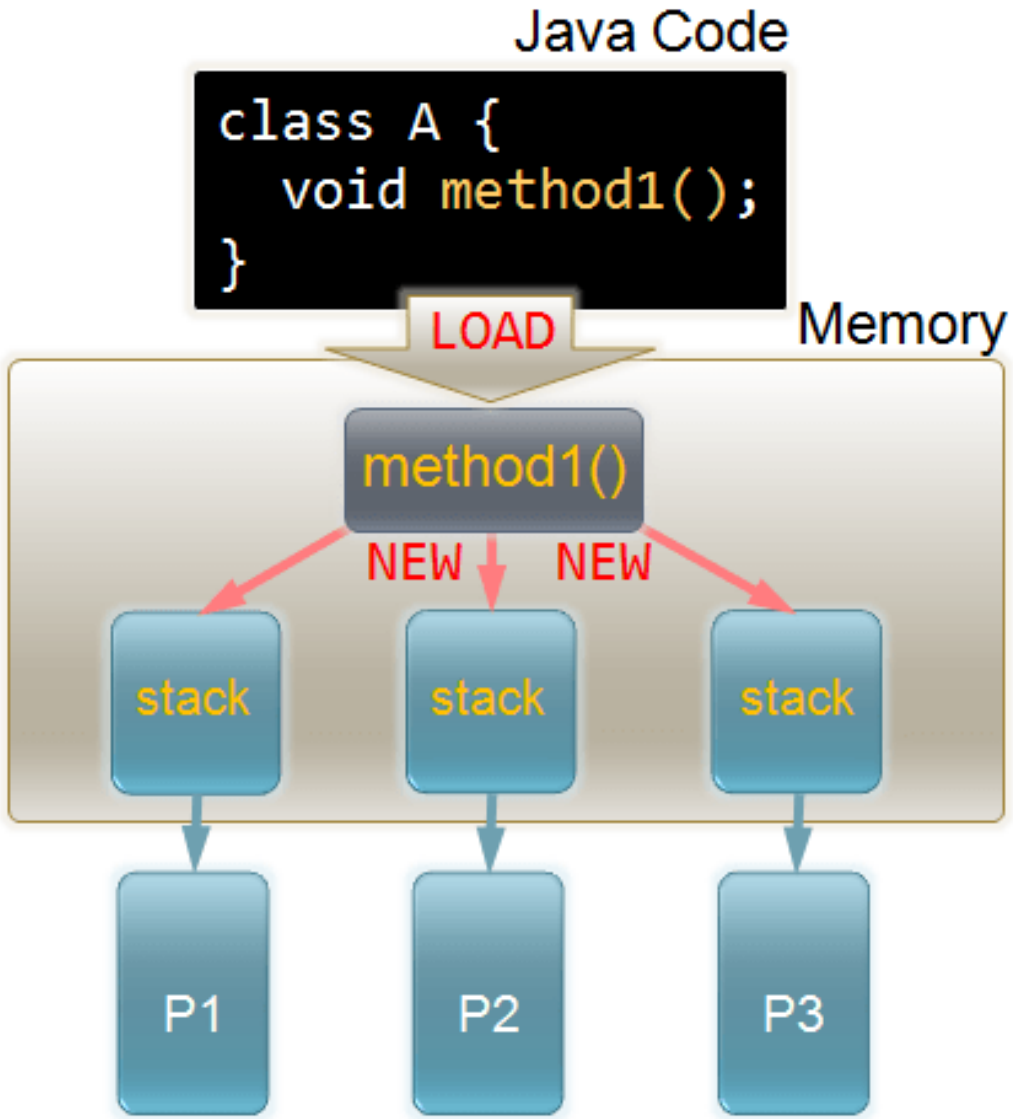
Instance Fields



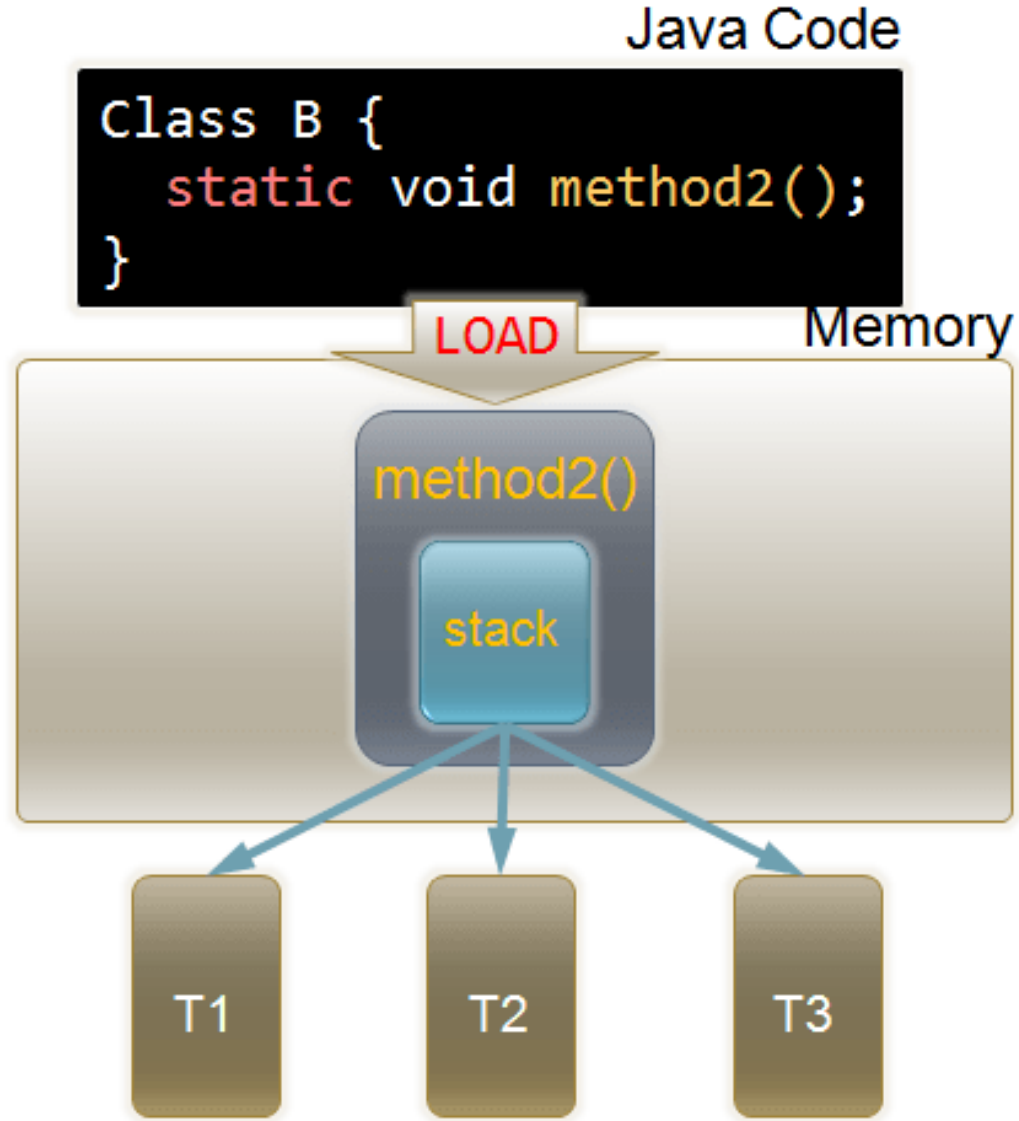
Static Fields



Instance Methods



Static Methods



CASE STUDY: test case for instance & static

```
public class TEST {

    int data;
    public static void main(String[] args) {
    TEST A = new TEST();
    TEST B = new TEST();
    A.data = 10;
    B.data = 20;
    System.out.println(A.data + B.data);
    }
}
```

```
public class TEST {

    static int data;
    public static void main(String[] args) {
    TEST A = new TEST();
    TEST B = new TEST();
    A.data = 10;
    B.data = 20;
    System.out.println(A.data + B.data);
    }
}
```

```
public class TEST {

    static int data;
    public static void main(String[] args) {
    TEST A = new TEST();
    TEST B = new TEST();
    A.data = 10;B.data = 20;
    int val = ++A.data+B.data++;
    System.out.println(val);
    }
}
```

```
public class TEST {

    int data;
    public static void main(String[] args) {
    TEST A = new TEST();
    TEST B = new TEST();
    A.data = 10;B.data = 20;
    int val = ++A.data+B.data++;
    System.out.println(val);
    }
}
```

Test:

```
public class TEST {

    static int data;
    public static void main(String[] args) {
    TEST A = new TEST();
    TEST B = new TEST();
    A.data = 10;B.data = 20;data = 100;
    int val = ++A.data+B.data++;
    System.out.println(val);
    }
}
```

```
public class TEST {
    static int data=100;
    public static void main(String[] args) {
    System.out.println(data);
    int data=200;
    System.out.println(data);
    }
}
```

```
public class TEST {
    static int data=100;
    public static void main(String[] args) {
    System.out.println(data);
    {int data=200;}
    System.out.println(data);
    }
}
```

Remove **static** int data?

```java
public class TEST {
    public static void Print() {
        System.out.println("Static method");
    }
    public static void main(String[] args) {
    Print();
    }
}
```

```java
2  public class TEST {
3      public void Print() {
4          System.out.println("Instance method");
5      }
6      public static void main(String[] args) {
7      Print();
8      }
9  }
```

```java
public class TEST {
    public void Print() {
        System.out.println("Instance method");
    }
    public static void main(String[] args) {
    TEST A = new TEST();
    A.Print();
    }
}
```

```java
public class TEST {
    public static void Print(int x,int y) {
        System.out.println(Add(x,y));
    }
    public static int Add(int a,int b) {
        return a+b;
    }
    public static void main(String[] args) {
    Print(10,20); //Static method call Static method
    }
}
```

```
2  public class TEST {
3      public static void Print(int x,int y) {
4          System.out.println(Add(x,y));
5      }
6      public int Add(int a,int b) {
7          return a+b;
8      }
9      public static void main(String[] args) {
10     Print(10,20); //Static method call Instance method
11     }
12 }
```

How to fix error?

```
2  public class TEST {
3      public void Print(int x,int y) {
4          System.out.println(Add(x,y));
5      }
6      public static int Add(int a,int b) {
7          return a+b;
8      }
9      public static void main(String[] args) {
10     Print(10,20); //Instance method call Static method
11     }
12 }
```

How to fix error?