

9

Fields

Introducing Fields

- › Variables that are declared in the class; stored in the objects.
- › Isolated for each object.

Objects



regNo: MHI23
carModel: Honda City
carYear: 2020



```
class Car
{
    string regNo;
    string carModel;
    int carYear;
}
```

Syntax of Field

accessModifier modifier type FieldName;

1. private
2. protected
3. private protected
4. internal
5. protected internal
6. public

1. static
2. const
3. readonly

Static fields are common to all objects.
Accessible with class name.

Const field's value can't be modified.
Compiler replaces all constant names with respective value.
Const is by default "static".

Readonly field's value can't be modified.
Compilation-time restriction only.

Access Modifiers of Fields

- › Access Modifier (a.k.a. "Access Specifier" or "Visibility Modifier") specifies the accessibility of fields, where the fields can be accessible; they provide security for the fields.

Access Modifier	In the same class	In the child classes at the same assembly	In the other classes at the same assembly	Child classes at other assembly	Other classes at other assembly
private	Yes	No	No	No	No
protected	Yes	Yes	No	Yes	No
private protected	Yes	Yes	No	No	No
internal	Yes	Yes	Yes	No	No
protected internal	Yes	Yes	Yes	Yes	No
public	Yes	Yes	Yes	Yes	Yes

Static Fields

- › Static fields are store outside the object.
- › Static fields are common to all objects of a class.

Class Memory in Heap

bankName: Bank of Dummyland

Objects in Heap

accountNumber: 1001
accountHolderName: Scott
currentBalance: 5000

accountNumber: 1002
accountHolderName: Bob
currentBalance: 6000

```
class BankAccount
{
    long    accountNumber;
    string  accountHolderName;
    double  currentBalance;
    static string bankName;
}
```

Instance Fields (vs) Static Fields

	Instance Fields	Static Fields
Storage	Stored in Objects	Stored in Class's memory.
Related to	Represents data related to objects.	Represents common data that belongs to all objects.

Declaration	Declared without "static" keyword. Syntax: type fieldName;	Declared with "static" keyword. Syntax: static type fieldName;
Accessible with	Accessible with object (through reference variable).	Accessible with class name only (not with object).
When memory gets allocated	Allocated separately for each object, because instance fields are stored "inside" the objects.	Allocated only once for the entire program; i.e. when the class is used for the first time while executing the program.

Constant Fields

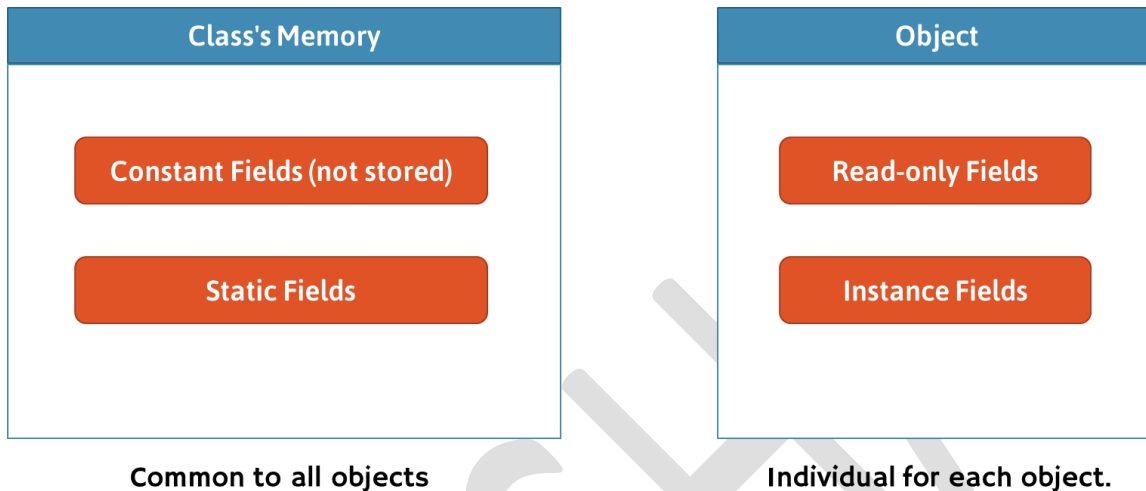
- › Constant Fields are like static fields, that are common to all objects of the class.
- › We can't change the value of constant field.
- › Constant Fields are accessible with class name [not with object].
- › Constant Fields are not stored in the object; will not be stored anywhere.
- › Constant Fields will be replaced with its value, while compilation; so it will not be stored anywhere in memory.
- › Constant Fields must be initialized, in line with declaration (with a literal value only).
- › Constants can also be declared as 'local constants' (in a method).

AccessModifier **const** **type** **FieldName = value;**

Readonly Fields

- › Readonly Fields are like instance fields, that is stored in every object, individually.
- › We can't change the value of readonly field.
- › Readonly Fields are accessible with reference variable [with object].
- › Readonly Fields must be initialized, either "in-line with declaration" [or] "in the constructor".

```
AccessModifier readonly DataType FieldName = value;
```



Key Points to Remember

- › Fields are variables that are declared in the class; but stored in objects.
- › Access modifiers of fields: private, protected, private protected, internal, protected internal, public
- › Modifiers of fields: static, const, readonly
- › Instance fields are individual for each object; Static fields are common (one-time) for all objects.
- › Constants must be initialized along with declaration; Readonly fields must be initialized either 'along with declaration' or in 'instance constructor'.