Memory Types

* Working set
* Stack
* Heap

Namespace

* Provides a virtual grouping thing for your .Net artifact (classes and delegates simply anything u create inside a .Net)
* A namespace can have more than one class but each class should have a unique name.

|  |
| --- |
| Notes\*   * Method has to be inside a class * Name of local and global variables can be same * When u access var in local it will access local not global variables. * A global var cannot be used inside a static method |

Code:

Int x = 100;

Int y = 200;

Int z = x;

Box firstBox = new Box();

firstBox.width = 12345;

Box.height = 100;

Box secondBox = new Box();

secondBox.width = 98765;

Stack Heap

New Box Width = 98765

New Box Width = 12345

secondBox

Box.height 100 100Second

firstBox

Z 100

Y 200

X 100

.Net Master class

* The base class to all classes in .Net is system.classes
* All classes inherit from system.objects
* Four important methods of system.object,
  + GetType()
  + GetHashCode()
  + Equals(object, object)
  + ToString()
* The child class can use override but parent cannot use override.
* Override – Specifies its own methods.

Working with classes

* Name of a class should be noun.
* Name of a class should start with upper case.
* A class can be public or internal (means friend) but it cannot be private or protected.
* Current class derived and other classes part of the same class can access friend or internal class.
* A class must have at least one constructor.
* A constructor is a very special method.
* The name of the class and constructor will have to be the same.
* If the programmer does not provide a constructor to the class, then the compiler will add a parameter less constructor to the class.
* If a parameterized constructor is defined in the class, then the compiler will not add the default constructor.
* A group of objects that share the same semantics and structure is a class.

Members of a class

* Field’s (also called as class variables)
* Properties
* Events
* Constructors
* Methods
* Inner Classes (Also called as aggregate)
* Finalizers (Acts as destructors) -----|| Avoid using it. It’s bad to have it. ||-------

Classification

* The process of grouping objects in a problem statement is called as classification.

Constructors

* Main purpose to make objects ready to use.
* To make members or object usable that it should initialize them to be ready to use.
* They doesn’t have a return type.

Notes\*

* The methods of a class show the behaviors of that class.
* **SOLID** principles
  + The **S**ingle Responsibility Principle
  + The **O**pen-Closed Principle
  + The **L**iskov Substitution Principle
  + The **I**nterface Segregation Principle
  + The **D**ependency Inversion Principle

Static Classes

* A class that has static cannot be instantiated.
* Static classes cannot have constructors.
* Java doesn’t have static classes, only C# has it.
* A static class can only have static variables or methods.
* It is illegal to have non static method inside a static class.
* Inheritance is not allowed, that is static classes cannot have children.
* Polymorphism and Inheritance are not supported in static classes.
* Static constructors don’t support access specifiers.

Private Constructors

* You cannot create objects directly for a class with a private constructor.
* A class with a private constructor cannot have children.
* A class marked as sealed (same as final in java) cannot have children.

Singleton

* At least one but not more than one is called singleton objects/classes.

Why OOPs?

* An object consists of,
  + Identifier
  + State
  + Behavior
* Access point is identifier.
* Identifier of object is known as reference.
* An object without any identifiers is called garbage or an orphan.
* Polymorphism is of two types,
  + Runtime Polymorphism
  + Design time Polymorphism

Interface

* Interface doesn’t create objects so it’s best suited for abstraction than an abstract class.