Essentials of Object Oriented Programming

Objectives

- On completion of this Session you will be able to
  - Name major pillars of Object Model
  - Write a class and create objects
  - Use constructor to initialize the object.
  - Use properties and indexers
  - Compare non static and static methods of class.
  - Overload methods & operators
  - Use exception handling
  - Write partial class

What is an object ?

- An entity which has well defined structure and behavior.
- Characteristics of an object are:
  - State
  - Behavior
  - Identity
  - Responsibility

Object - Bank Account

Attributes	State	Behavior	Identity	Responsibility
Balance	Balance = 7000	Open Balance Withdraw Report	Account Number = 4141	Keeps a track of a stores money with the facility of deposits and withdraw
Interest Rate	Interest Rate = 6%			
Account number	Account Number= 4141			

Object Oriented Programming

- Pillars of Object Oriented Systems
  - Major
    - Abstraction
    - Encapsulation
    - Inheritance
  - Minor
    - Typing
    - Concurrency
    - Persistence

Abstraction

- Abstraction is the process of identifying the key aspects of an entity and ignoring the rest.
- Only Domain Expertise can do right abstraction.
- Abstraction of Person object
  - Useful for social survey
  - Useful for health care industry
  - Useful for Employment Information

Encapsulation

- Encapsulation is process of compartmentalizing the element of an Abstraction that constitute its structure and behavior.
  - Serves to separate interface of an abstraction and its implementation.
  - User is aware only about its interface; any changes to implementation does not affect the user.

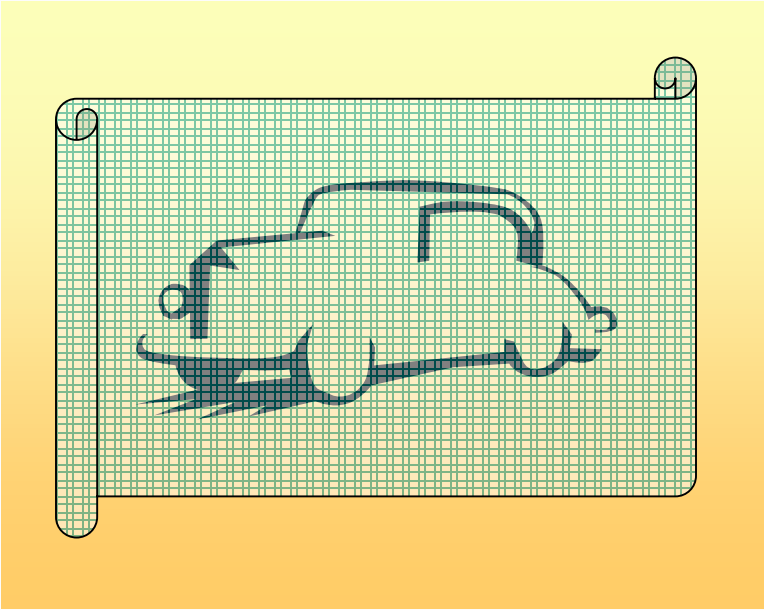
Inheritance

- Classification helps in handling complexity.
- Factoring out common elements in a set of entities into a general entity and then making it more and more specific.
- Hierarchy is ranking or ordering of abstractions.

Minor pillars

- Typing
  - Is the enforcement of the entity, such, that objects of different types may not be interchanged, or at the most, they may be interchanged only in very restricted ways.
- Concurrency
  - Different objects responding simultaneously.
- Persistence
  - Persistence is the property of an object through which its existence transcends time and/or space.

Objects and Classes



Blue print (class)

Object A

Object B

Class - Members

```
public class Employee
{
   int _id;
   string _name;
   public static int _count;
   public Employee()
   {
    //Initializtion
   }
   public Employee(int id,
                   string name)
   {
    _id=id;
    _name=name;
   }
   public ~Employee()
   {
    //DeInitializtion
   }
}
```

- Class has members
  - Fields
  - Methods
    - Constructors & Destructor
    - User defined methods
  - Properties & Indexers

```
static  void Main()
{
    Employee e1 = new Employee( );
    Employee e2=new Employee(123, "John");
}
```

Constructor

- Implicit method gets invoked during object creation.
- Member Initialization is feature of constructor.
- More than one constructors could be written.

```
class Employee
{ public Employee()
  {//Initializtion
  }
  public Employee(int id,string name)
  {
       _id=id;
       _name=name;
   }
```

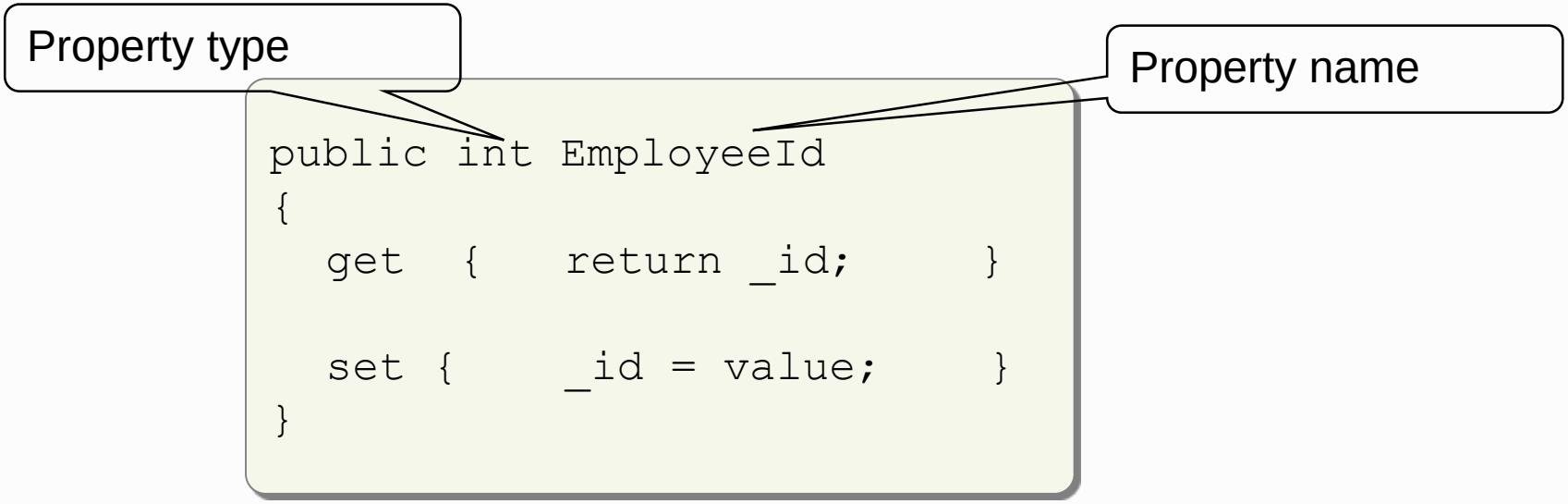
Destructor

- Implicit method that gets invoked by CLR before object destruction.
- Member De-Initialization is feature of destructor.

```
public ~Employee()
{
        //DeInitializtion
}
```

Properties

- Known as smart fields.
- Special methods that assign and retrieve values from the underlying data member.
- Have two accessors:
  - `get` - retrieves data member values.
  - `set` - enables data members to be assigned.



Indexer

- Known as Smart Array

```
public class IndexierClass{
private int [] myArray = new int[100];
public int this [int index]
{
  get
   {
     // Check the index limits.
     if (index < 0 || index >= 100)
         return 0;
      else
         return myArray[index];
     }
  set
   {
   if (!(index < 0 || index >= 100))
          myArray[index] = value;
    }
}
}
```

```
public static void
   Main()
{
IndexerClass b = new
       IndexerClass();
b[3] = 256;
}
```

Non - Static method

- Also known as Instance method.
- Method is invoked using object reference.

```
public class Employee
{ . . . // all members
   public float CalculateBonus()
    {     //logic defined       }
}
```

```
public static void Main()
{
Employee e = new Employee();
float bonus  = e.CalculateBonus( );
}
```

Static Member and Static Methods

- Data to be shared by all objects of the class is stored in static data members.
- Only single copy exists per class.

```
public class Employee
{
   static int count;
   static int ShowCount()
   {
     return count;
   }
}
```

```
public static void Main()
{
int numberOfEmployees=
   Employee.ShowCount();
Console.WriteLine(numberOfEmployees);
}
```

Method Overloading

- Overloading is the ability to define several methods with the same name, provided each method has a different signature.

```
public class MathEngine
{
      public static double FindSquare(double number)
      {
         //logic defined
       }

     public static double FindSquare(int number)
      {
      //logic defined
      }

}
```

```
public static void Main()
{
   double res=MathEngine.FindSquare(12.5);
   double num= MathEngine.FindSquare(12);
}
```

Operator Overloading

- Giving additional meaning to existing operators.
- Technique that enhances power of extensibility.

```
public static Complex operator+(Complex c1,Complex c2)
{
  Complex temp = new Complex ();
  temp.real = c1.real + c2. real;
  temp.imag = c1.imag + c2.imag;
  return temp;
}
```

```
static void  Main(){
Complex o1 = new Complex(2, 3);
Complex o2 = new Complex(3, 4);
Complex o3 = o1 + o2;
Console.WriteLine(o3.real + "---" + o3.imag);
}
```

Operator Overloading Restrictions

- Following operators cannot be overloaded.

Conditional logical	&&,||
Array indexing operator	[]
Cast Operators	()
Assignment operators	+=,-=,*=,/+ etc
	=,., ?: ,->, new ,is ,size of, typeof

- The comparison operators, if overloaded, must be overloaded in pairs .
  - If == is overloaded then != must also be overloaded.

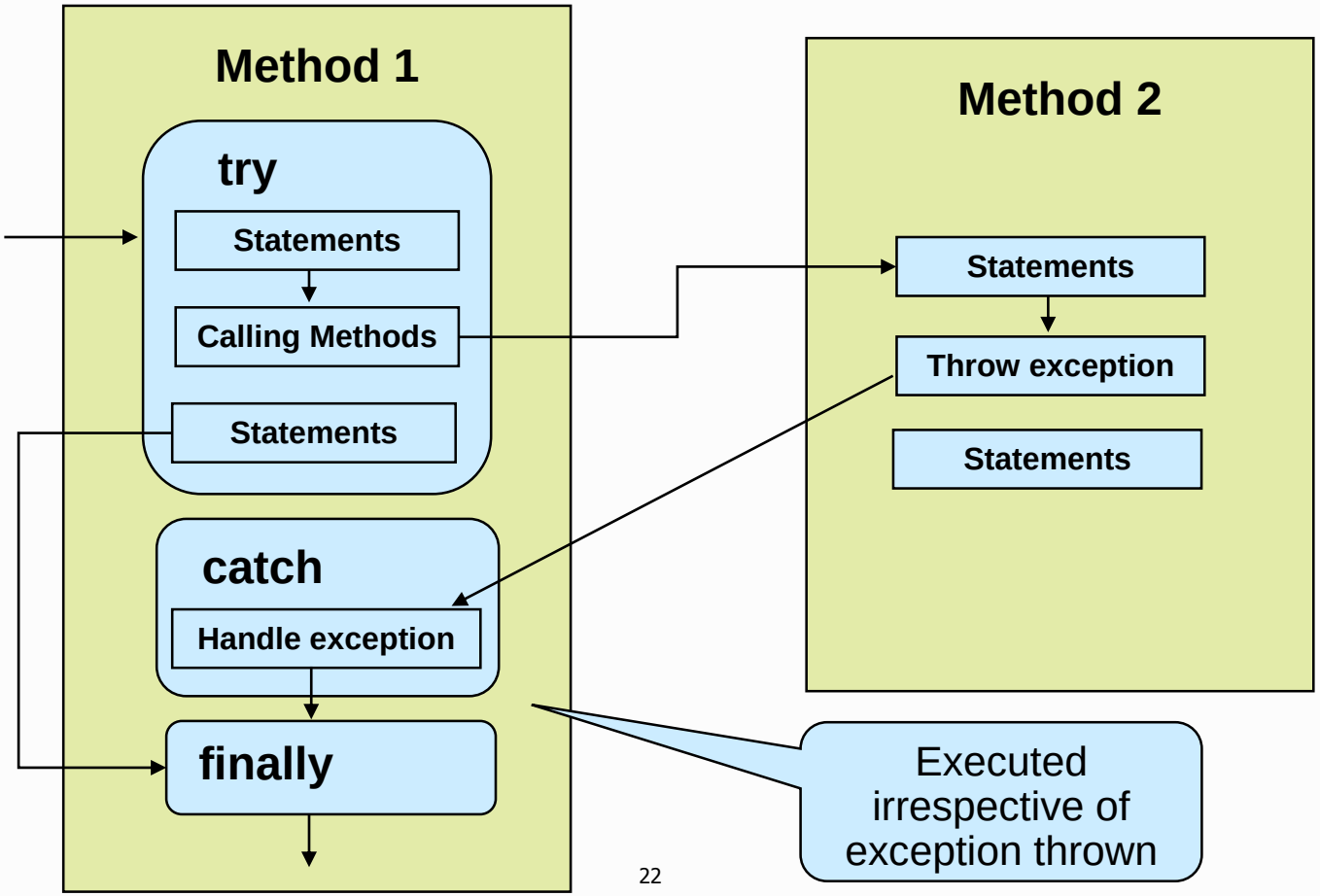
Exceptions

- Abnormalities that occur during the execution of a program (runtime error).
- .NET framework terminates the program execution for runtime error.
  - e.g. Divide by zero, Stack overflow, File reading error, loss of network connectivity

```
int a, b = 0 ;
Console.WriteLine( "My program starts " ) ;
try {
      a = 10 / b;
    }
catch ( Exception e ) {
      Console.WriteLine (e.Message) ;
     }
Console.WriteLine ( "Remaining program" ) ;
```

Exception Handling

- Mechanism to detect and handle runtime errors.



.NET Exception classes

- SystemException
- ArgumentException
- ArgumentNullException
- ArithmeticException
- ArrayTypeMismatchException
- CoreException
- DivideByZeroException
- FormatException
- IndexOutOfRangeException
- InvalidCastExpression
- InvalidOperationException
- NullReferenceException
- OutOfMemoryException
- StackOverflowException

Partial Class

- A class can be spread across multiple source files using the keyword `partial`.
- All source files for the class definition are compiled as one file with all class members.
- Access modifiers used for defining a class should be consistent across all files.

Quick Recap…

- Writing class maps the two major pillars of Object Model-abstraction and encapsulation.
- An object is an instance of a class that occupies memory and has a finite lifespan.
- Constructors are methods which initialize an object.
- Properties help to get or set the value of data members.
- Methods with same name and different signatures are said to be overloaded.
- Operator Overloading defines existing operators to perform operations on user defined types.
- Exception are runtime errors which are handled using `try – catch` block.