UML TOOLS

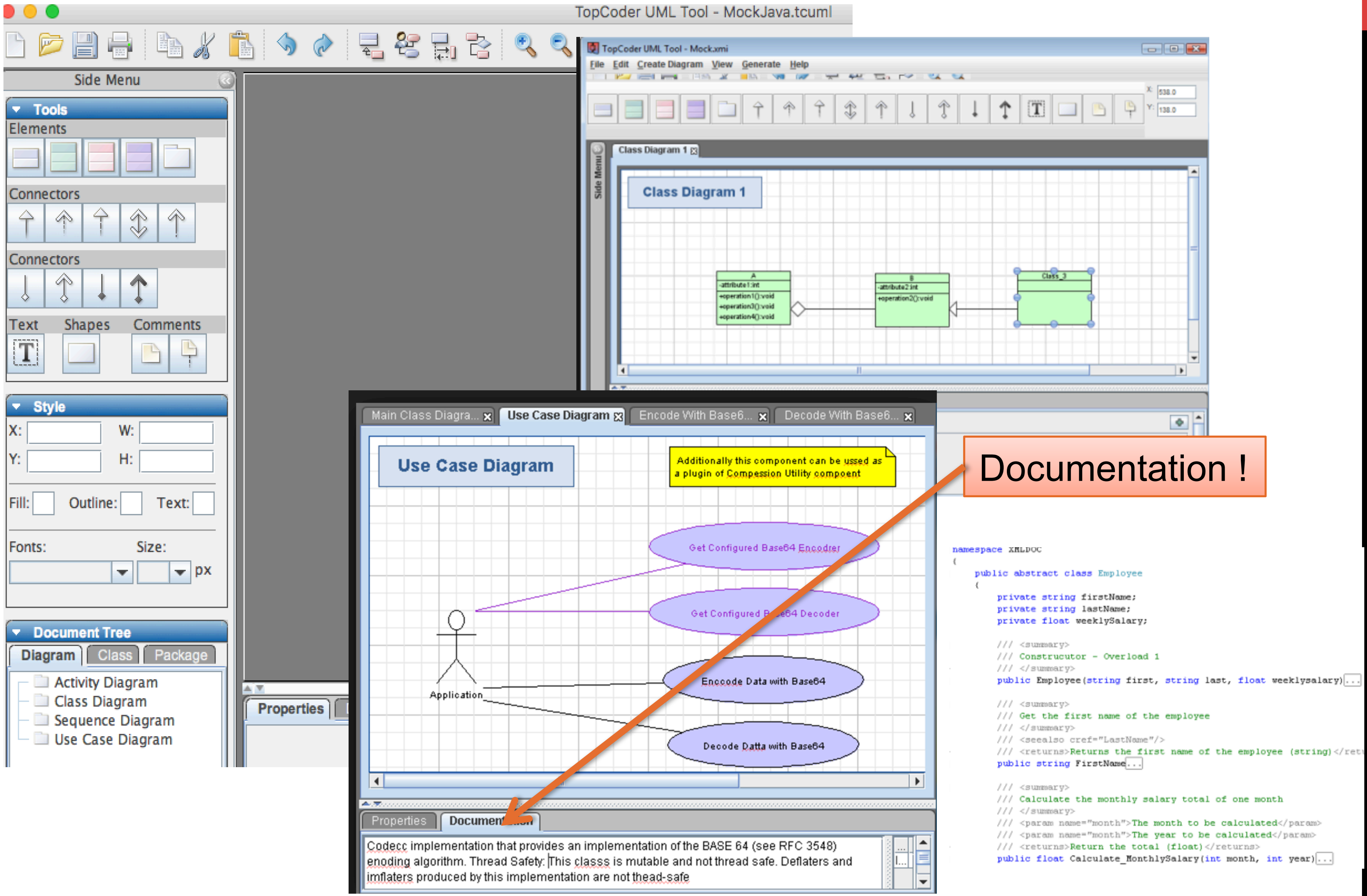
(PART 2)

BY PAKITA SHAMOI

SOME OF THE OPTIONS…

- ArgoUML
- Astah
- TopCoder UML Tool
- Modelio
- StarUML
- NClass
- Umbrello UML Modeller
- Most reputable IDEs also have UML (e.g., Eclipse, Netbeans, Visual Studio, etc.)
- Many, many others…

TOP CODER UML TOOL

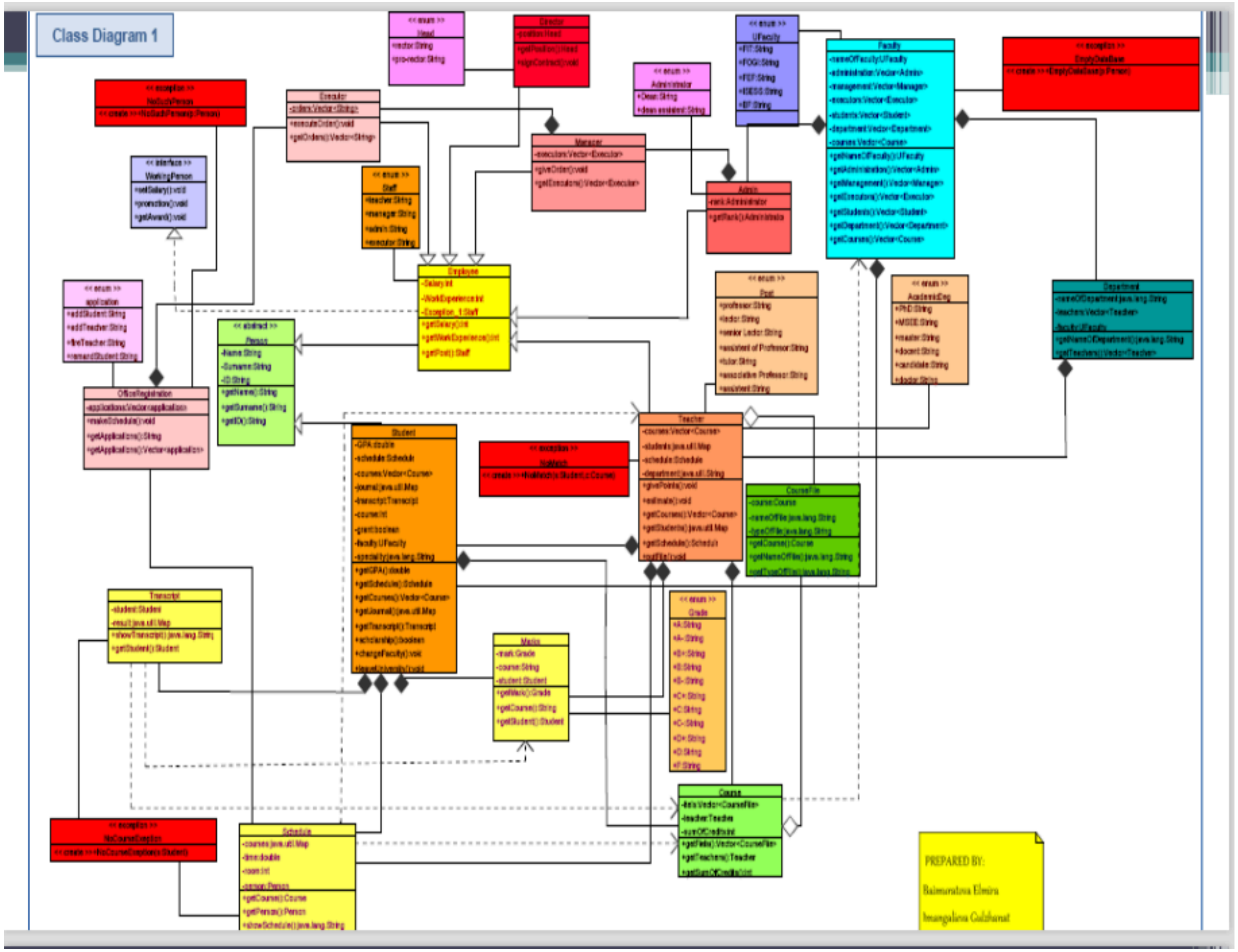


Codecc implementation that provides an implementation of the BASE 64 (see RFC 3548) enoding algorithm. Thread Safety: This classs is mutable and not thread safe. Deflaters and imflaters produced by this implementation are not thead-safe

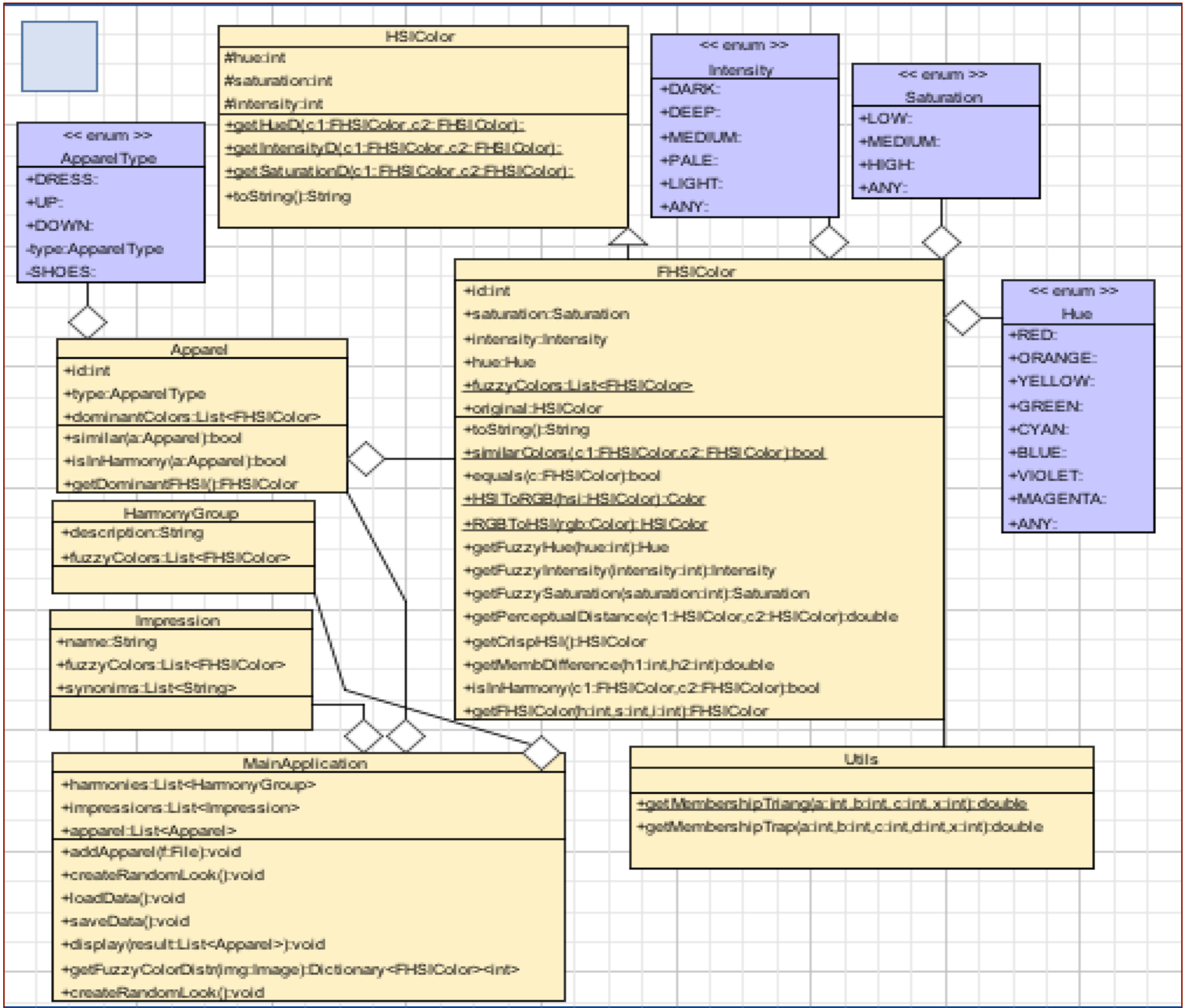
Documentation !

```
namespace XMLDOC
{
    public abstract class Employee
    {
        private string firstName;
        private string lastName;
        private float weeklySalary;

        /// <summary>
        /// Construcutor - Overload 1
        /// </summary>
        public Employee(string first, string last, float weeklysalary)...

        /// <summary>
        /// Get the first name of the employee
        /// </summary>
        /// <seealso cref="LastName"/>
        /// <returns>Returns the first name of the employee (string)</ret
        public string FirstName...

        /// <summary>
        /// Calculate the monthly salary total of one month
        /// </summary>
        /// <param name="month">The month to be calculated</param>
        /// <param name="month">The year to be calculated</param>
        /// <returns>Return the total (float)</returns>
        public float Calculate_MonthlySalary(int month, int year)...
```

EXAMPLES

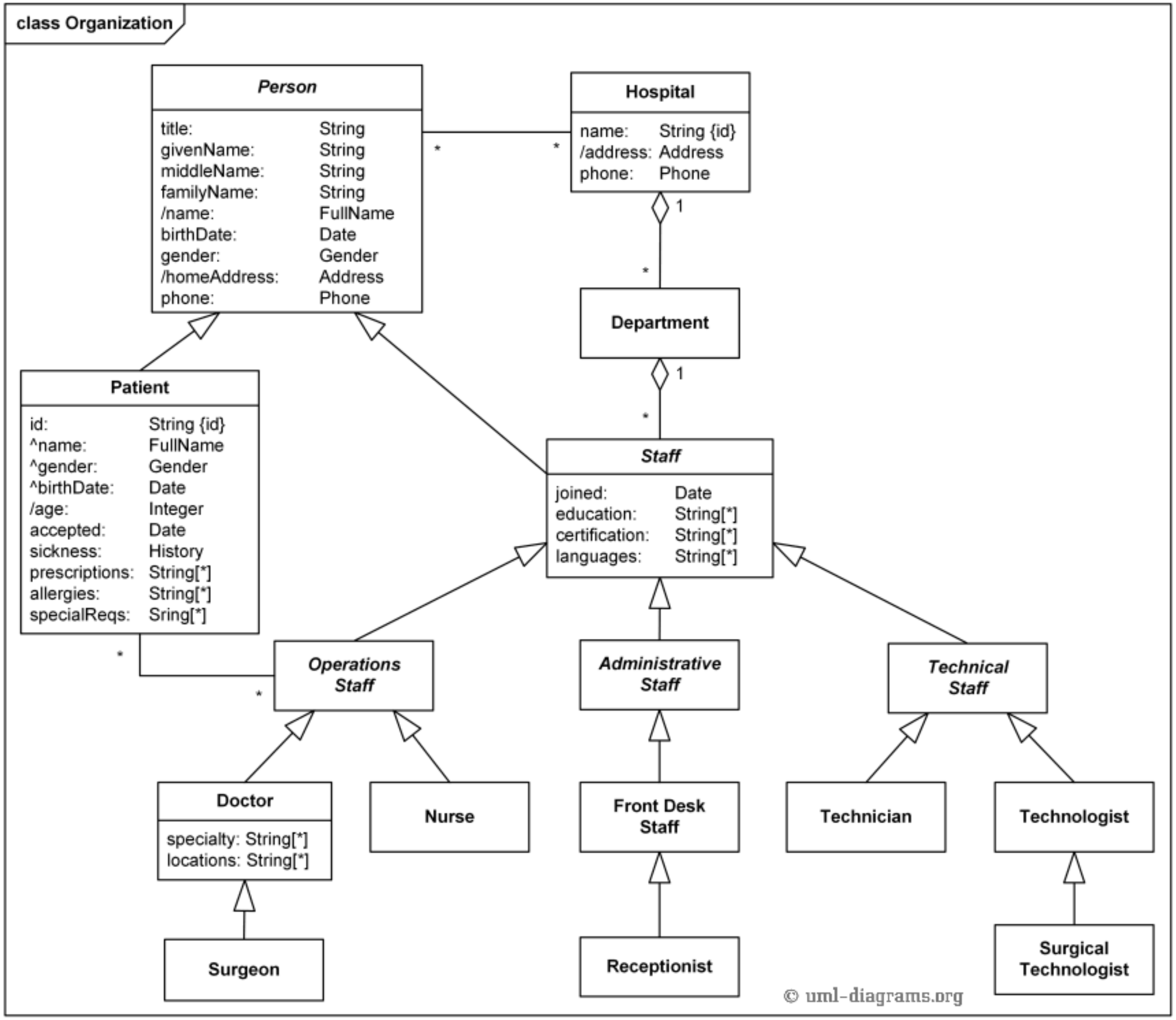


Class Diagram 1
<< enum >> Head
Director
<< exception >> NoSuchPerson
Executor
Manager
<< enum >> Administrator
<< enum >> UFaculty
Faculty
<< exception >> EmptyDataBase
<< interface >> WorkingPerson
<< enum >> Staff
Admin
Employee
<< enum >> application
<< abstract >> Person
<< enum >> Post
<< enum >> AcademicDeg
Department
OfficeRegistration
Student
<< exception >> NoMatch
Teacher
CourseFile
Transcript
Marks
<< enum >> Grade
Course
<< exception >> NoCourseException
Schedule
PREPARED BY:
Baimuratova Elmira
Imangalieva Gulzhanat

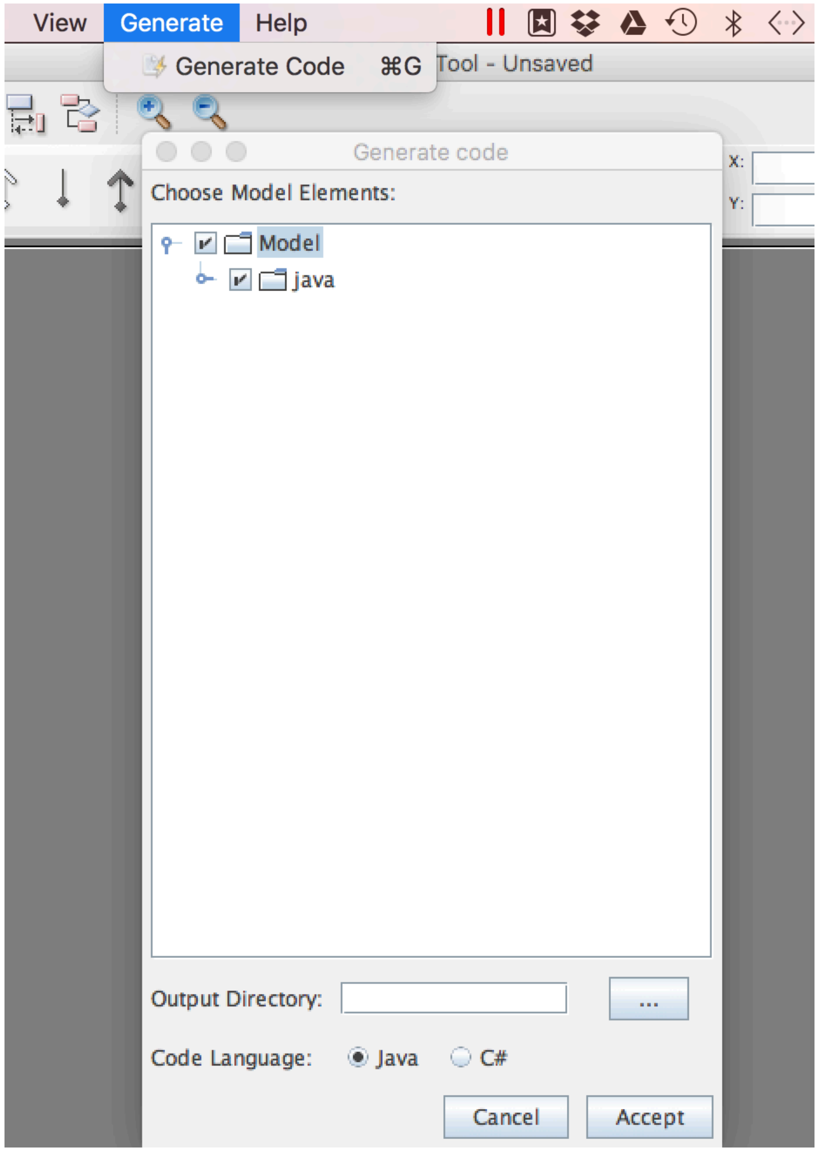


HSIColor
#hue:int
#saturation:int
#intensity:int
+getHueD(c1:FHSIColor,c2:FHSIColor):
+getIntensityD(c1:FHSIColor,c2:FHSIColor):
+getSaturationD(c1:FHSIColor,c2:FHSIColor):
+toString():String
<< enum >>
Intensity
+DARK:
+DEEP:
+MEDIUM:
+PALE:
+LIGHT:
+ANY:
<< enum >>
Saturation
+LOW:
+MEDIUM:
+HIGH:
+ANY:
<< enum >>
ApparelType
+DRESS:
+UP:
+DOWN:
-type:ApparelType
-SHOES:
FHSIColor
+id:int
+saturation:Saturation
+intensity:Intensity
+hue:Hue
+fuzzyColors:List<FHSIColor>
+original:HSIColor
+toString():String
+similarColors(c1:FHSIColor,c2:FHSIColor):bool
+equals(c:FHSIColor):bool
+HSIToRGB(hsi:HSIColor):Color
+RGBToHSI(rgb:Color):HSIColor
+getFuzzyHue(hue:int):Hue
+getFuzzyIntensity(intensity:int):Intensity
+getFuzzySaturation(saturation:int):Saturation
+getPerceptualDistance(c1:HSIColor,c2:HSIColor):double
+getCrispHSI():HSIColor
+getMembDifference(h1:int,h2:int):double
+isInHarmony(c1:FHSIColor,c2:FHSIColor):bool
+getFHSIColor(h:int,s:int,i:int):FHSIColor
<< enum >>
Hue
+RED:
+ORANGE:
+YELLOW:
+GREEN:
+CYAN:
+BLUE:
+VIOLET:
+MAGENTA:
+ANY:
Apparel
+id:int
+type:ApparelType
+dominantColors:List<FHSIColor>
+similar(a:Apparel):bool
+isInHarmony(a:Apparel):bool
+getDominantFHSI():FHSIColor
HarmonyGroup
+description:String
+fuzzyColors:List<FHSIColor>
Impression
+name:String
+fuzzyColors:List<FHSIColor>
+synonims:List<String>
MainApplication
+harmonies:List<HarmonyGroup>
+impressions:List<Impression>
+apparel:List<Apparel>
+addApparel(f:File):void
+createRandomLook():void
+loadData():void
+saveData():void
+display(result:List<Apparel>):void
+getFuzzyColorDistr(img:Image):Dictionary<FHSIColor><int>
+createRandomLook():void
Utils
+getMembershipTriang(a:int,b:int,c:int,x:int):double
+getMembershipTrap(a:int,b:int,c:int,d:int,x:int):double

EXAMPLE – HOSPITAL MANAGEMENT



GENERATING A CODE



Choose Model Elements:

- [✓] Model
  - [✓] java

Output Directory:

Code Language: [● Java] ○ C#

Suppose you created a class diagram and generated the code. As you start the implementation, you, for sure, will need to change some fields or methods or even add a new class.
How to do it? ---→ Reverse engineering

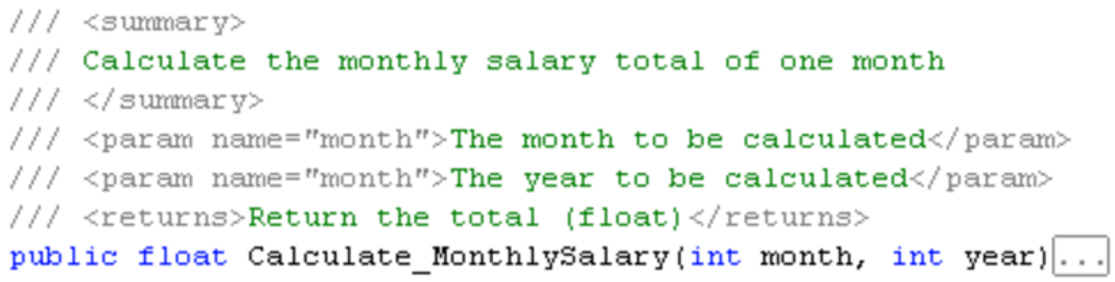
DOCUMENTATION

- It is important to provide adequate documentation for all of your applications.
- Provide enough comments to enable a developer who was not involved in creating the original application to follow and understand how the application works.
- Visual Studio allows you to write documentation and easily integrate it to IDE.
- Majority of UML Tools allow you to prepare documentation at a design stage

GENERATING DOCUMENTATION

- You can use C# comments to generate XML documentation for your applications.

- Documentation comments begin with three forward slashes (///) followed by an XML documentation tag.



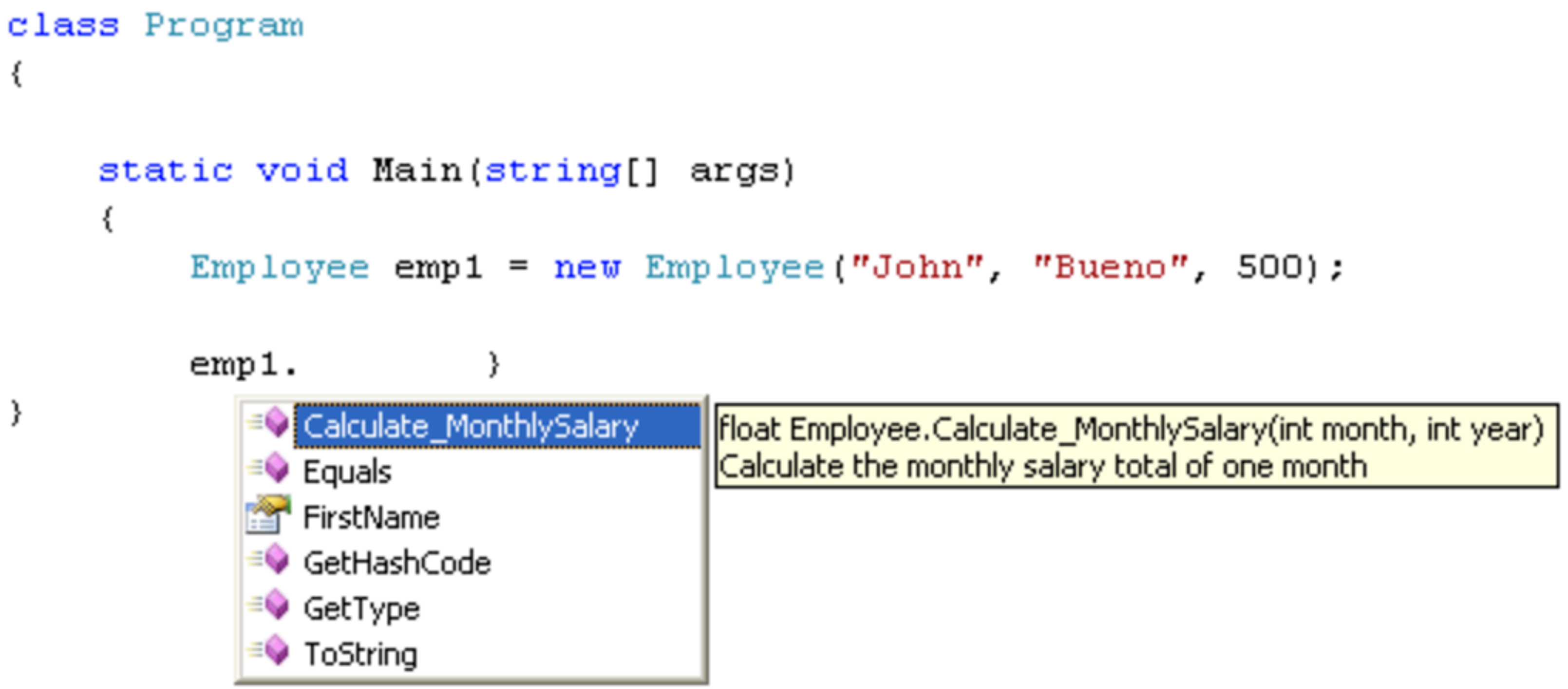
```
/// <summary>
/// Calculate the monthly salary total of one month
/// </summary>
/// <param name="month">The month to be calculated</param>
/// <param name="month">The year to be calculated</param>
/// <returns>Return the total (float)</returns>
public float Calculate_MonthlySalary(int month, int year)...
```

- You can compile the XML tags and documentation into an XML file by using the C# compiler with the /doc option:

```
csc myprogram.cs /doc:mycomments.xml
```

- If there are no errors, you can view the XML file that is generated by using a tool such as Internet Explorer.

RESULT



```
class Program
{

    static void Main(string[] args)
    {
        Employee emp1 = new Employee("John", "Bueno", 500);

        emp1.          }
}
```

float Employee.Calculate_MonthlySalary(int month, int year)
Calculate the monthly salary total of one month