

GEOG 178/258

Week 6:

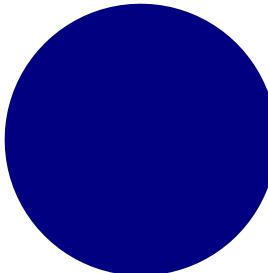
UML, Poisson Distributions & Midterm Prep

mike johnson



PART 1: UML



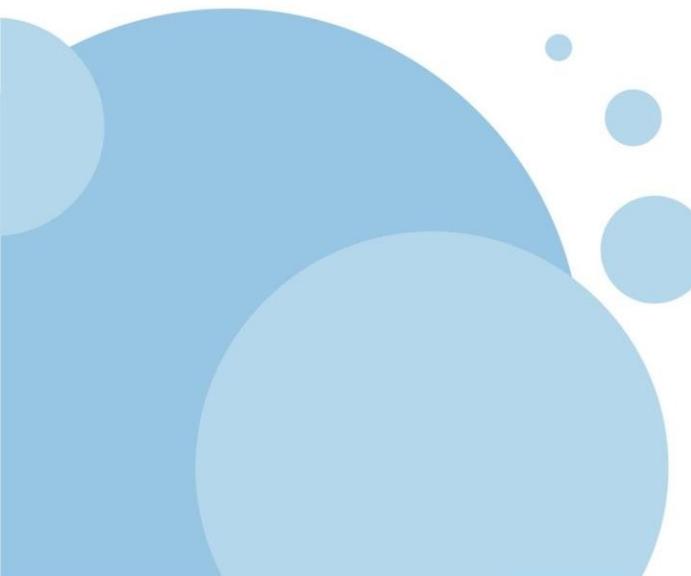


Unified Modeling Language

Week

6

UML



UML is a **standardized** modeling language consisting of diagrams

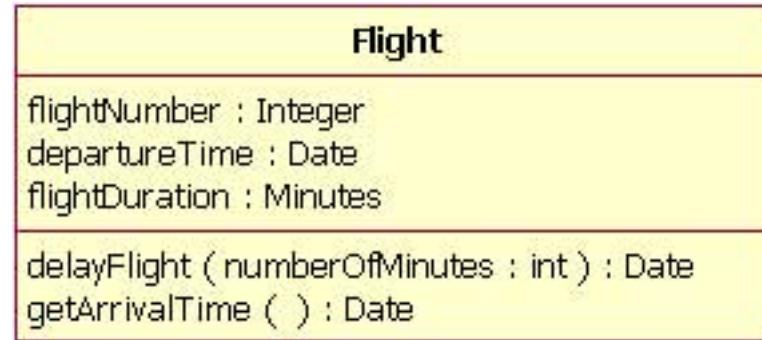
1. Developed to help system and software developers specify, visualize, document and construct software systems.
2. UML is another way of modeling an abstraction of reality

UML Class

Week

6

UML



Classes are represented as rectangles with stacked compartments:

The top compartment shows the **class name** (Flight)
The middle: the **class attributes**
The last: the class operations (aka methods)

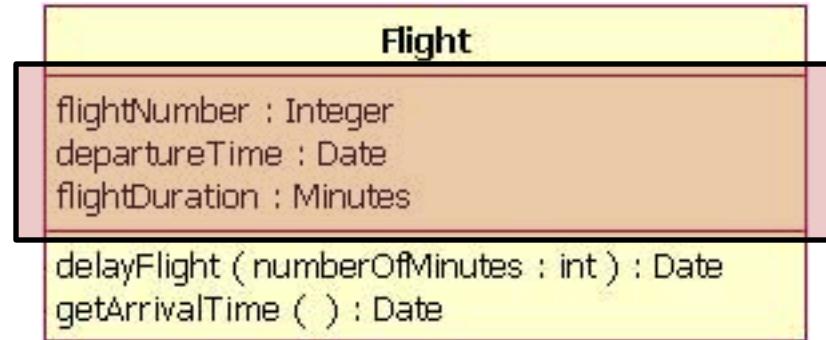
Think about how this already mirrors our structure of (**Attributes, Constructors, Getters& Setters, Methods**)

UML Class

Week

6

UML



Attribute lines are optional but if included are written in the following structure:

Name : attribute type

In many “everyday” class diagrams, the attribute types usually show units that make sense to readers (i.e., minutes, dollars, etc.). However, a class diagram that will be used to generate code needs classes whose attribute types are limited to the types provided by the programming language, or types included in the model that will also be implemented in the system.

Often default values will be provided as well:

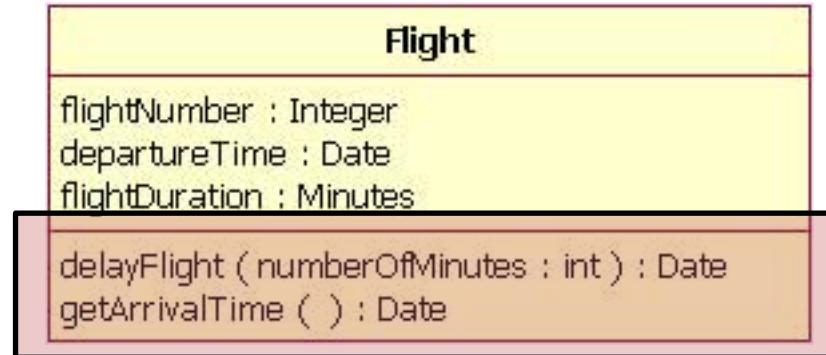
MyBank: double = 0

UML Class

Week

6

UML



Operations are documented in a list format in the following notation:

Name(parameter list) : type of value returned

(think to the signature of your methods like `isInside!`)

When parameters are needed the name and type should be explicitly provided:

isInside (P1 : Point, P2: Point) : Boolean

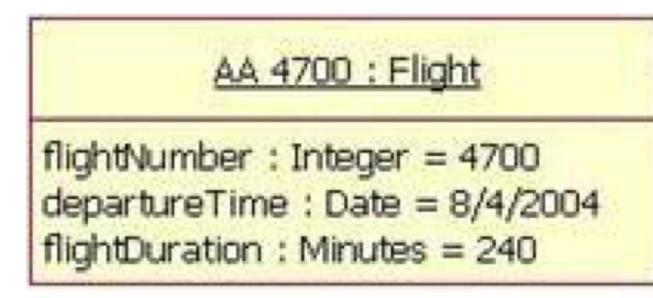
UML Class

Week

6

UML

In important (or specific cases) UML can be used to diagram a particular instance of a class

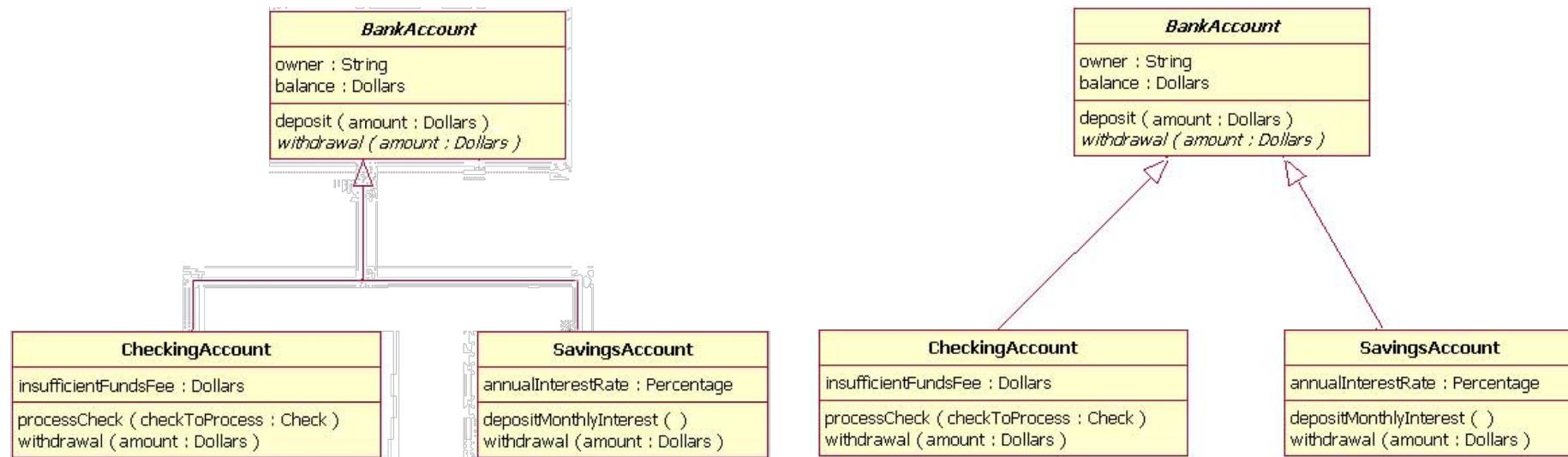


UML Inheritance

Week

6

UML



Symbol Descriptions

Week

6

UML

	<i>Generalization</i>		<i>For Public</i>
	<i>Inheritance</i>		
	<i>Composition</i>		<i>For Private</i>
	<i>Aggregation</i>		<i>For Protected</i>
	<i>Dependencies</i>		<i>For Derived</i>
	<i>Properties</i>		
	<i>Multiplicity</i>		<i>For Package</i>

[For more look here](#)

UML Class Generic Example

Week

6

UML

```
public class Example {  
    private int x;  
    protected int y;  
    public int z;  
  
    public Example() { ... }  
  
    public String toString() { ... }  
    private void foo(int x) { ... }  
    protected int bar(int y, int z) { ... }  
}
```

Example
-x:int #y:int +z:int
+«constructor»Example() +toString():String -foo(x:int) #bar(y:int,z:int):int

OGC Point

Week

6

UML

Private variables

Geometry
Point
+ X() : Double
+ Y() : Double
+ Z() : Double
+ M() : Double

Figure 4: Point

6.1.4.2 Methods

- **X():Double** — The *x*-coordinate value for *this* Point.
- **Y():Double** — The *y*-coordinate value for *this* Point.
- **Z():Double** — The *z*-coordinate value for *this* Point, if it has one. Returns NIL otherwise.
- **M():Double** — The *m*-coordinate value for *this* Point, if it has one. Returns NIL otherwise.

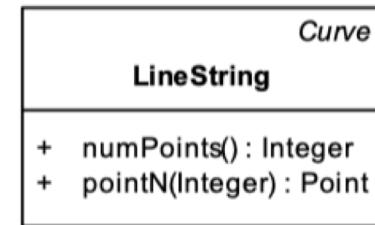


Figure 7: LineString

6.1.7.2 Methods

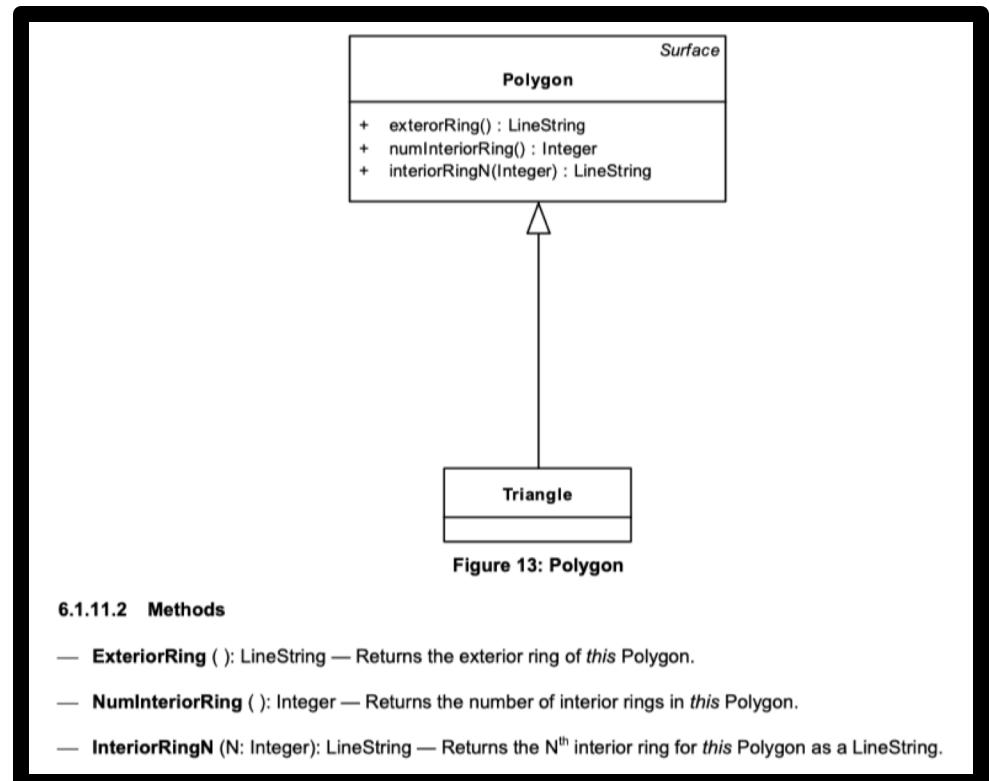
- **NumPoints ()**: Integer — The number of Points in *this* LineString.
- **PointN (N: Integer)**: Point — Returns the specified Point N in *this* LineString.

OGC Polygon

Week

6

UML

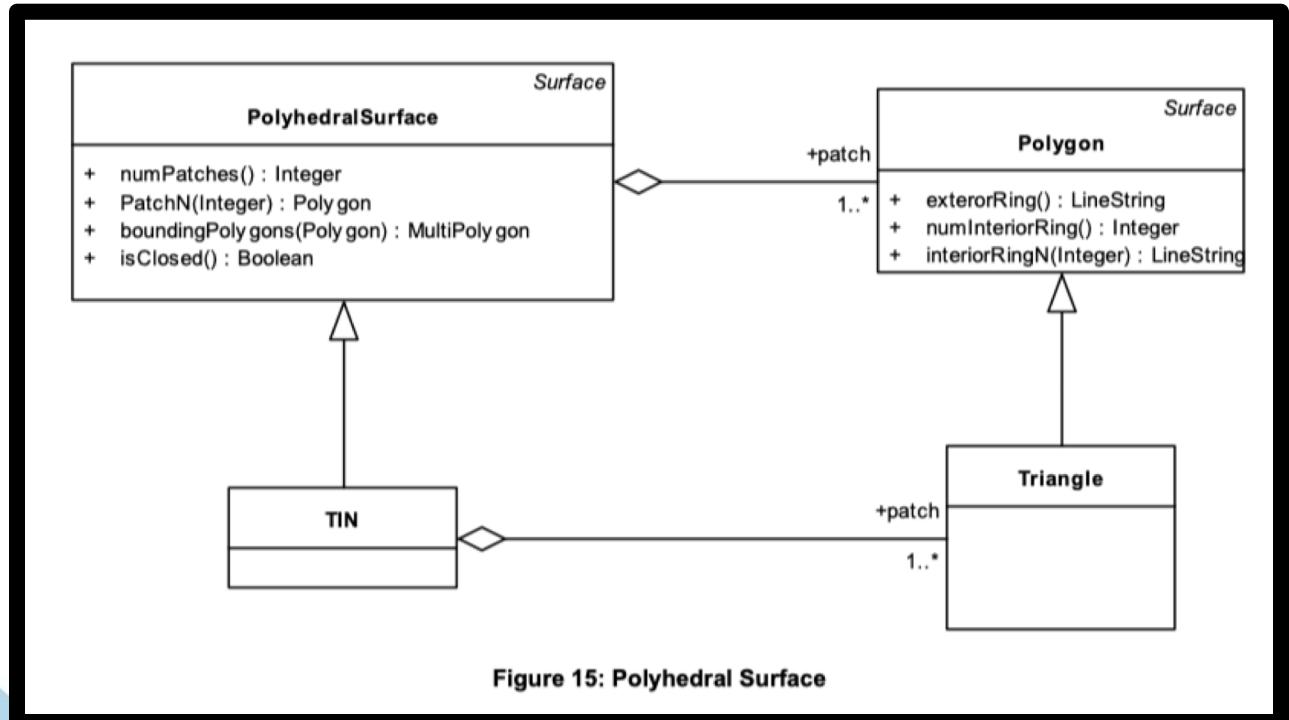


OGC. PolyhedralSurface

Week

6

UML



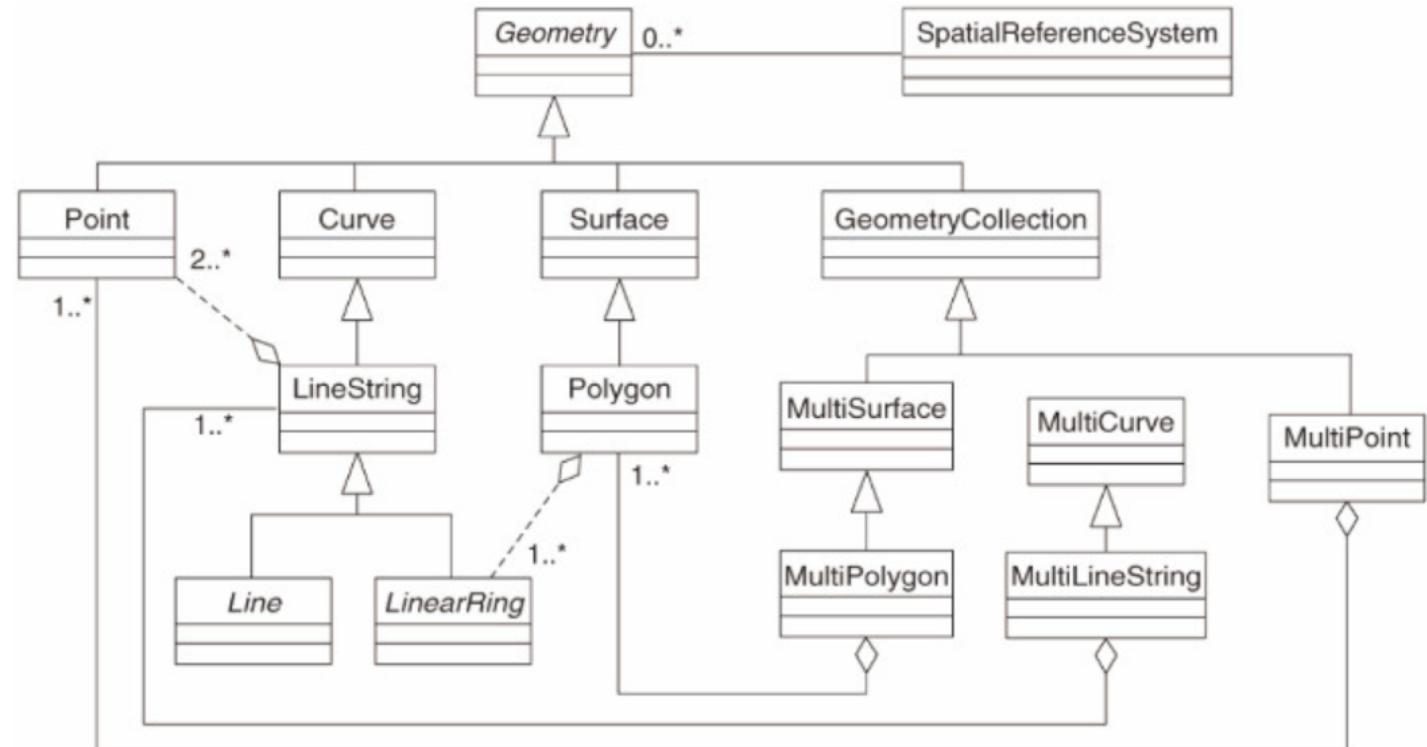
OGC Simple Features

Week

6

UML

OGC Simple Feature Access



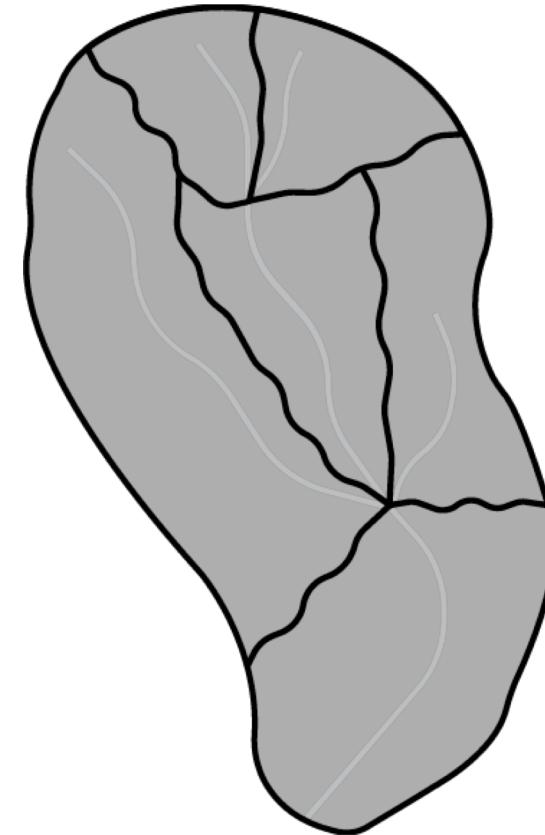
Real Life UML Data Modeling

Week

6

UML

WaterML (click me)

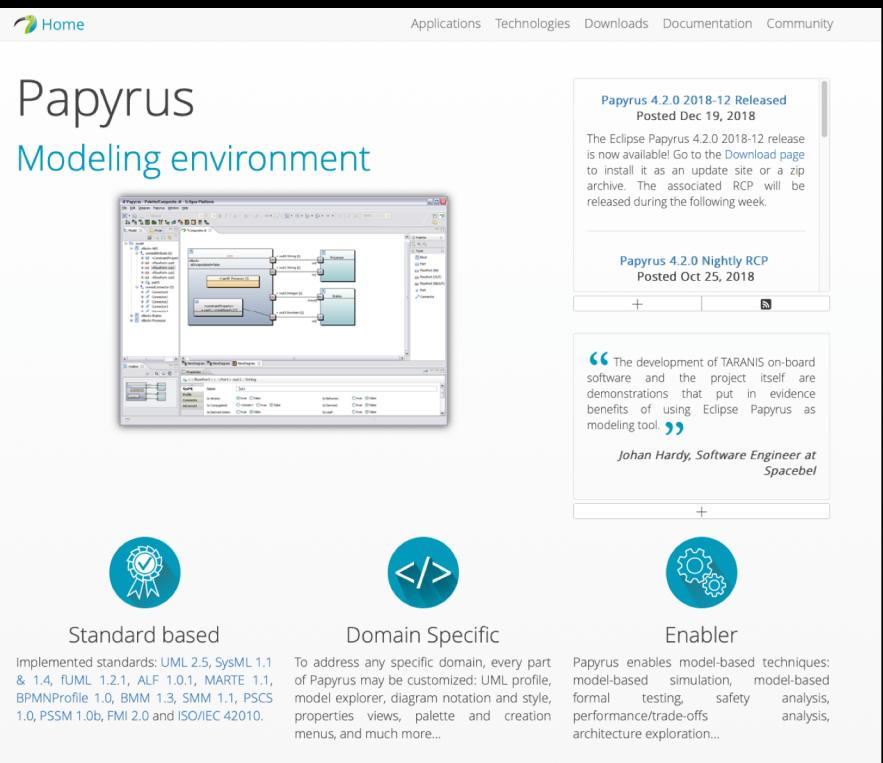


UML in Eclipse

Week

6

UML



The screenshot shows the Eclipse Papyrus interface. At the top, there's a navigation bar with links to Home, Applications, Technologies, Downloads, Documentation, and Community. Below the navigation bar, the title "Papyrus Modeling environment" is displayed. In the center, a UML class diagram is shown with various classes and associations. To the right of the diagram, there's a sidebar with news items. The first item is "Papyrus 4.2.0 2018-12 Released" posted on Dec 19, 2018. It describes the release as available for download as an update site or a zip archive. The second item is "Papyrus 4.2.0 Nightly RCP" posted on Oct 25, 2018. Below the sidebar, a quote from "Johan Hardy, Software Engineer at Spacebel" is displayed, stating: "The development of TARANIS on-board software and the project itself are demonstrations that put in evidence benefits of using Eclipse Papyrus as modeling tool." At the bottom of the page, there are three circular icons with text below them: "Standard based" (represented by a ribbon icon), "Domain Specific" (represented by a '</>' icon), and "Enabler" (represented by a gear icon). Descriptions for each category mention the implementation of specific UML profiles and the ability to customize the tool for different domains and enable model-based techniques.

Papyrus 4.2.0 2018-12 Released
Posted Dec 19, 2018

The Eclipse Papyrus 4.2.0 2018-12 release is now available! Go to the [Download page](#) to install it as an update site or a zip archive. The associated RCP will be released during the following week.

Papyrus 4.2.0 Nightly RCP
Posted Oct 25, 2018

“ The development of TARANIS on-board software and the project itself are demonstrations that put in evidence benefits of using Eclipse Papyrus as modeling tool. ”

Johan Hardy, Software Engineer at Spacebel

Standard based

Domain Specific

Enabler

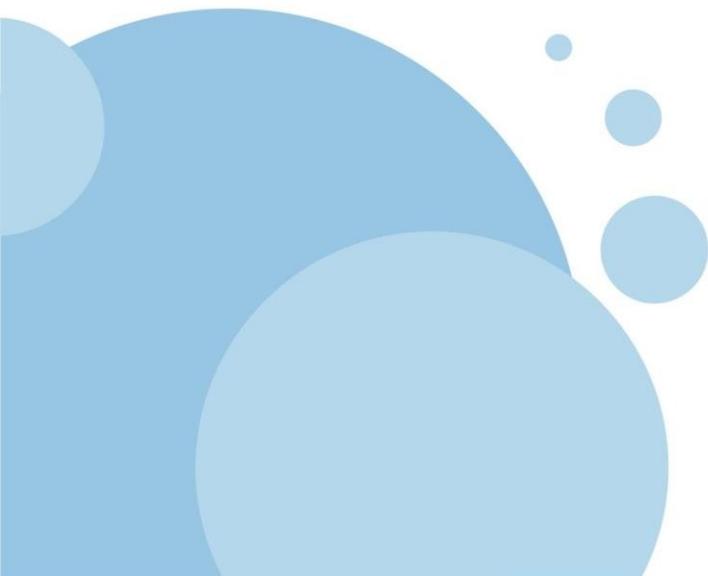
Implemented standards: UML 2.5, SysML 1.1 & 1.4, fUML 1.2.1, ALF 1.0.1, MARTE 1.1, BPMNProfile 1.0, BMM 1.3, SMM 1.1, PSCS 1.0, PSSM 1.0b, FMI 2.0 and ISO/IEC 42010.

To address any specific domain, every part of Papyrus may be customized: UML profile, model explorer, diagram notation and style, properties views, palette and creation menus, and much more...

Papyrus enables model-based techniques: model-based simulation, model-based formal testing, safety analysis, performance/trade-offs analysis, architecture exploration...

<https://www.eclipse.org/papyrus/>

PART 2: Poisson Distributions



Poisson Distribution

Week

6

Poisson

The Poisson Distribution is a **discrete** probability distribution that expresses the **probability** that a given number of events, occurring in a **fixed interval** of time or space with a **known constant rate** and **independently of the time since the last event**.

$$P = \frac{\text{average}^i}{i! \times e^{\text{average}}}$$

Story

Week

6

Poisson



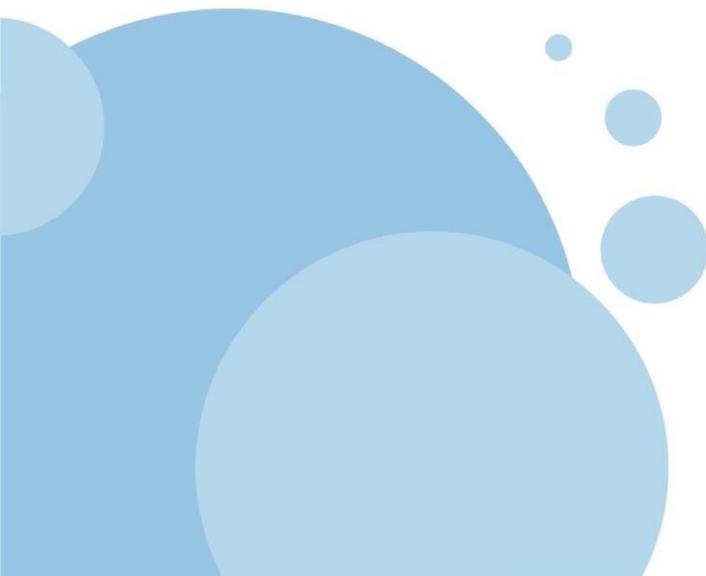
Number of Campers	Number of Plots
1	278
2	92
3	25
4	4
5	0
6	0
7	1

Our goal:

1. What is the Poisson Distribution of these campers ?
2. Can campers/plots able to be described by such a discrete distribution?

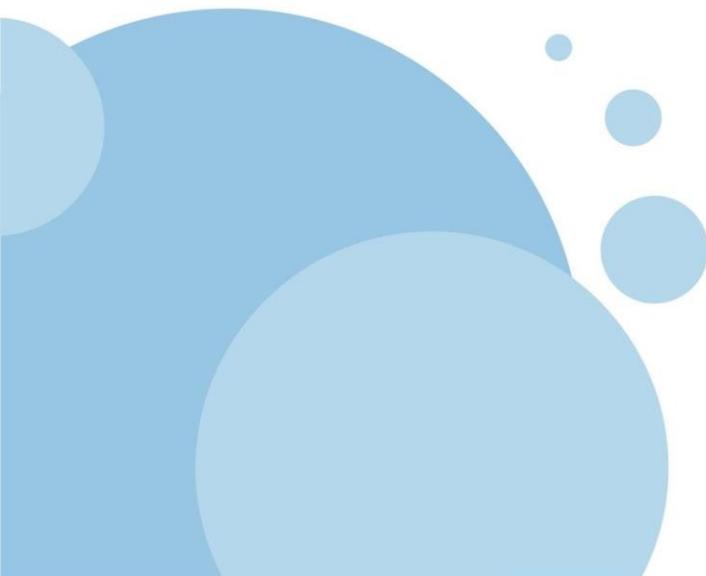
EXAMPLE # 1

Campers, Plots and Poisson



EXAMPLE # 2

Getting Poisson to Github

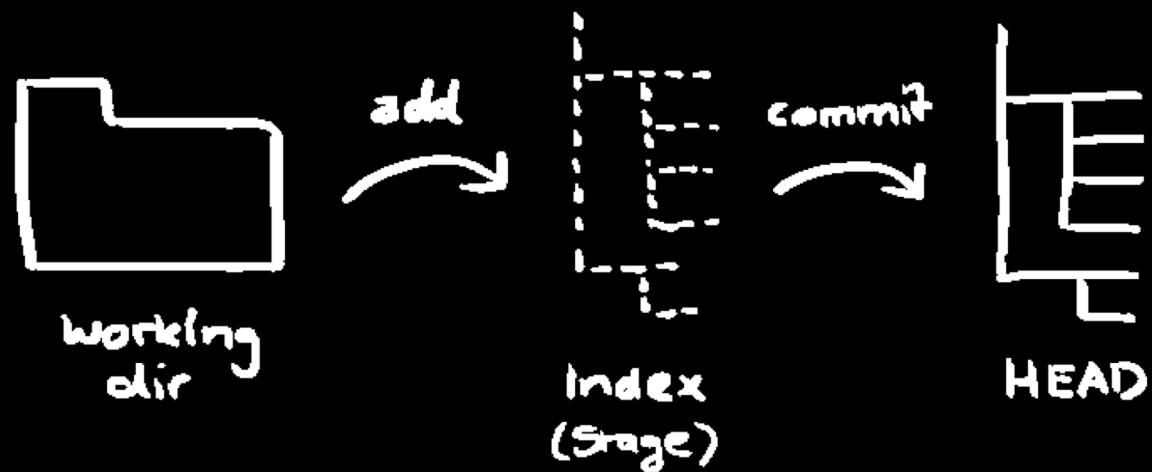


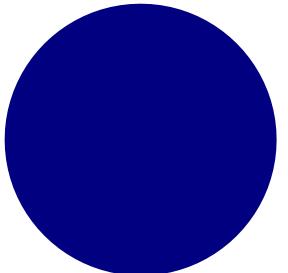
Version Control

Week

6

Github





Github Workflows

Week

6

Cheat sheet

Create a new git repository: *git init*

Checkout an existing Repo: *git clone <path>*

Connecting to remote repo: *git remote add origin <server path>*

Adding files: *git add <filename>*

Adding all files: *git add **

Committing to HEAD: *git commit -m message*

Pull files: *git pull origin master* *master can be subbed for any branch

Pull files and realign: *git pull --rebase*

Push files: *git push origin master* *master can be subbed for any branch

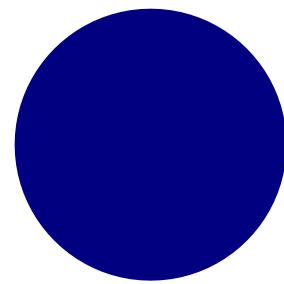
Branching

Week

6

Github





Week

6

Cheat sheet

Create a new branch called “new_feature”:

```
git checkout -b new_feature
```

Switch back to master:

```
git checkout master
```

Switch back to branch

```
git checkout new_feature
```

Combine branches (from master)

```
git checkout master *be sure your in master  
git merge new_feature
```

** Sometimes conflicts will occur between branches that make merging impossible.
You have to fix these manually and add them back in via git add <file>

*** If you really mess up a file you can get the original back:
`git checkout <filename>`

PART 3: HW and MT Hints



Homework Hints!

Week

6

Homework Hints

Interfaces

Geometry {getLength, getArea}

BoundingArea {isInside}[G]

Polypoints{getPoints, setPoints, getPointCount} [G]

Needed Classes:

BoundingBox (BA)

Circle (G)

Point (G)

PointBuffer (BA)

Polygon (PP)

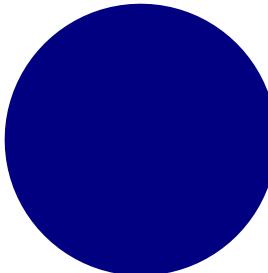
Polyline (PP)

Rectangle (G)

Square (G)

Test (NA)

() implements
[] extends



Midterm Examples

Week

6

MT

Check out the class site for an example set of MT questions

