You should all have a team by now.

For Monday, on a sheet of paper

- Name of team members, with email
- Name of the team (be creative, or I chose for you)
- A number between 0 and 100
- Course conflict with on Monday/Wednesday between 10h30 and 12h00.
Divide a large tasks in smaller components.
Easier to complete smaller components individually.

Dividing into subproblems
- Subproblems can be solved independently.
- Solutions to subproblems can be combined to solve the whole problem.

Cooking supper example
What are the components of Naval Battle?
Naval Battle (2)

Units
Terrain
Actions
Rules
Players
What is O.O. programming?

- A computer programming paradigm.
- Emphasizes the following
  - Abstraction
  - Information/Implementation hiding (encapsulation)
  - Modularity
Abstraction

simplify different things and treat them as the same

- Decomposition by changing the level of detail to be considered.
- Forget information and consequently to treat different things as if they were the same.

  - Files on a hard disk
  - Units in a game
Abstraction / Type Hierarchy

- A type family is defined by a type hierarchy.
- At the top of the hierarchy is a supertype that defines behavior common to all family members.
- Other members are subtypes of this supertype.
- A hierarchy can have many levels.
Example of Abstraction Hierarchy

- Optical
  - CD
  - DVD

- Media

- Magnetic
  - Disc
  - Tape
Example of Abstraction Hierarchy

- Media
  - Optical
    - CD
    - DVD
  - Magnetic
    - Disc
    - Tape
Info. / Implementation hiding

separating implementation from interfaces

- When observing an encapsulation, we can have two points of view:
  - From the outside (public view)
  - From the inside (private view)

- The advantages of a good encapsulation is the separation of the private and public views.
In a video game, how can I store the direction a player is facing?
How do I store the direction a player is facing?

- An integer?
  - 4 possible values: 1=North, etc
  - Values from 0 to 360?
- A float?
  - Values from 0 to 99.9?
- A character?
  - n,s,e and w?
- 4 booleans?
  - north, south, east, west?
How do I hide this from the user?

- `IsFacingNorth() : boolean`
- `IsFacingSouth() : boolean`
- `IsFacingEst() : boolean`
- `IsFacingWest() : boolean`
- `GetDegreeFacing() : int`
- `GetDirectionFacing() : int`
- Never allow other class to directly access your attribute.
- Once an attribute is public, it can never be changed.
  - Ex: img.pixeldData
- Make your attributes available using get/set methods.
  - this.connectionStatus Bad!
  - this.getConnectionStatus() Good!
public interface Point {
    public set(int x, int y);
    public int getX();
    public int getY();
}
Inside, point could be using Cartesian or Polar coordinates.

- Cartesian coordinates are more efficient when dealing with lots of translations.
- Polar coordinates are more efficient when dealing with lots of rotations.
Modularity

- Break down elements into units depending on themes and concerns.
- Minimizing interaction between these units improves maintainability.
Modularity in Mammoth
Specifications in English

- Company XYZ is a manufacturing company that produces cartoon figurines for big entertainment companies.
- This company needs an inventory and tracking system.
- The inventory system keeps track of how many of each figurines is stored in each warehouse.
- Figurines are stored in cases.
- Clients order the figurines and the cases are eventually shipped to clients.
This time, in UML
Unified Modeling Language (UML)

- A language, both graphical and textual, used throughout the entire process of project design (from requirements analysis to deployment).
- Semi-formal specification that captures structure of O.O.D.
- A standard tool for communicating a design.
Unified Modelling Language (UML)

-many notations

COLLECTION

structural
  class diagrams
  package diagrams

(object) interaction
  sequence diagrams
  collaboration diagrams

behaviour
  activity diagrams
  state charts

implementation
deployment
<table>
<thead>
<tr>
<th>Class Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
</tr>
<tr>
<td>Methods</td>
</tr>
<tr>
<td>User-defined constraints (invariants)</td>
</tr>
</tbody>
</table>

Constraints may also be written as note
### 2DPoint

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>x: int</td>
<td><code>getx()</code>: int { return x }</td>
</tr>
<tr>
<td>y: int</td>
<td><code>setx(a: int): void { x = a }</code></td>
</tr>
<tr>
<td></td>
<td><code>gety()</code>: int { return y }</td>
</tr>
</tbody>
</table>
- Classes are static, depict the design and structure at design-time

- Objects are dynamic and are instantiated (from a class) at run-time, they have state
Attributes vs Variables

- Attributes are considered at design-time, are some abstractly defined property.
- Variables are considered at implementation-time, are concretely defined properties.
<table>
<thead>
<tr>
<th><strong>objectName</strong>: ClassName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable = defaultValue</td>
</tr>
</tbody>
</table>