UPPSALA
UNIVERSITET

Instructions

UPPSALA
UNIVERSITET

Analysis and Design.

focus on Object orientation and Modular programming.

Refactoring

The main purpose of refactoring is to fight technical debt. It transforms a mess into clean code and simple design

https://refactoring.guru/refactoring/smells

The Postit method

A way to discover packages

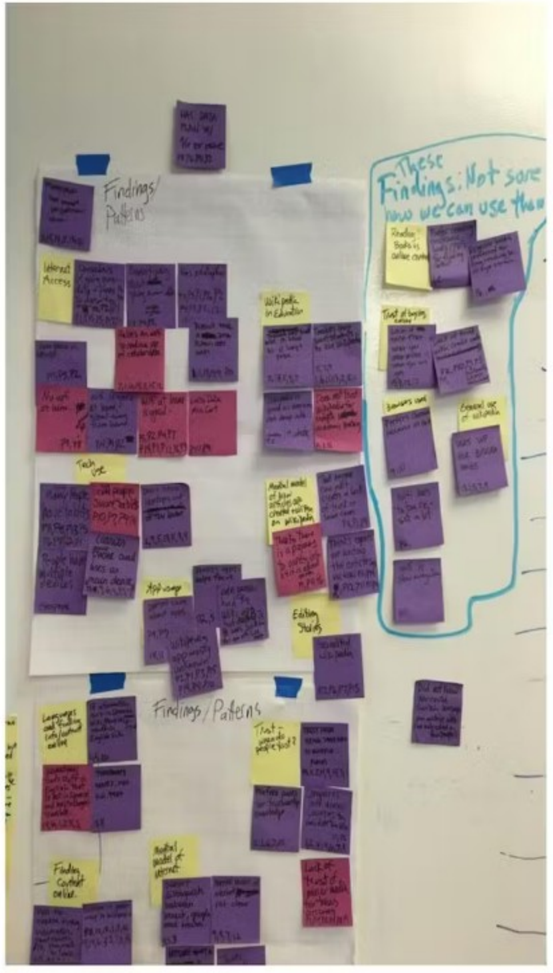
Write your Behaviours/functional requirements on a note, group similar behaviour in clusters, then once all your behaviours are sorted you can draw circles on on the white board delimiting your packages. This process can be done iteratively at any stage of the process.

If you are under FDA review remember to hide your sticky notes :-)

Miro boards https://miro.com/app/dashboard/ (lots of design templates)

ZOOM has a white board function that lets you do this virtually and in a meeting

Example of other online whiteboard https://ideaflip.com/



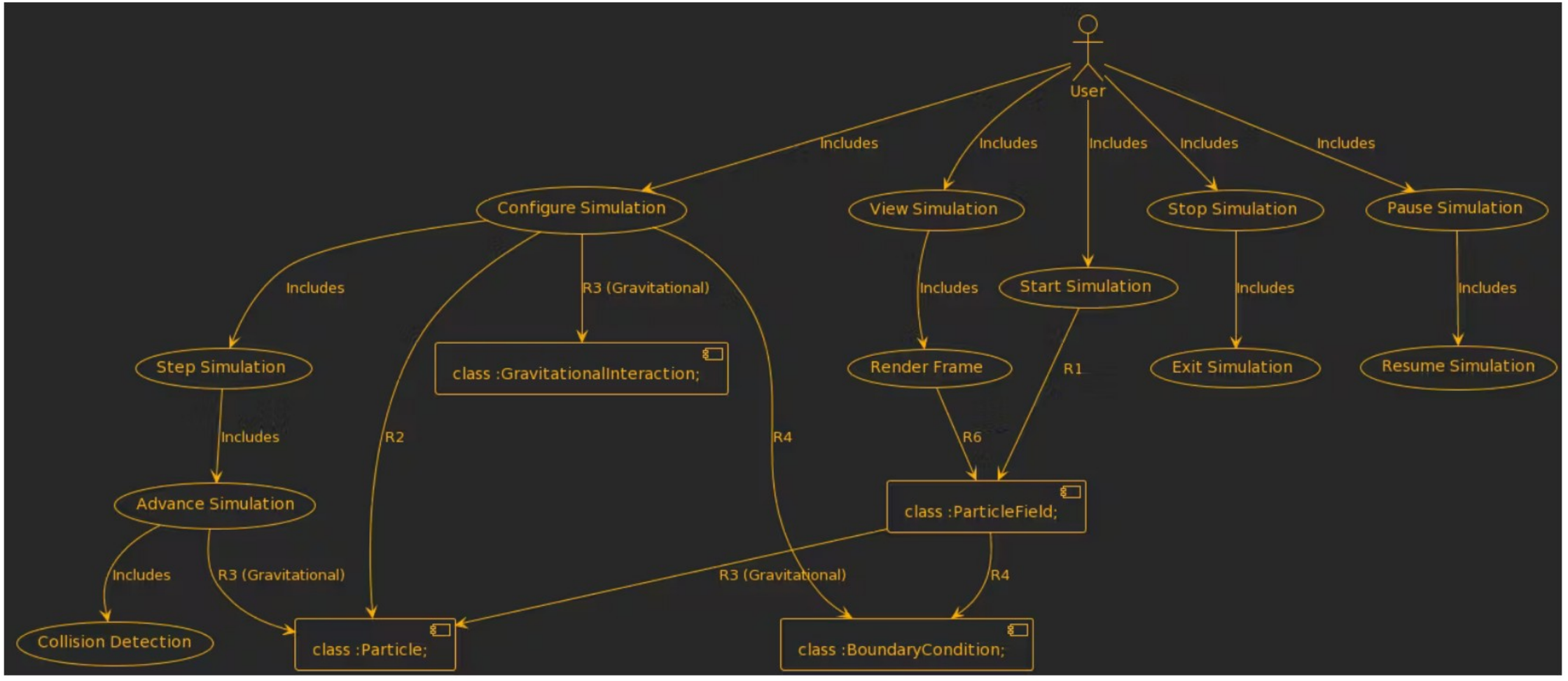
UPPSALA UNIVERSITET

Miro boards cannot currently be shown when exporting presentations.

Usecase inception

2

UPPSALA
UNIVERSITET



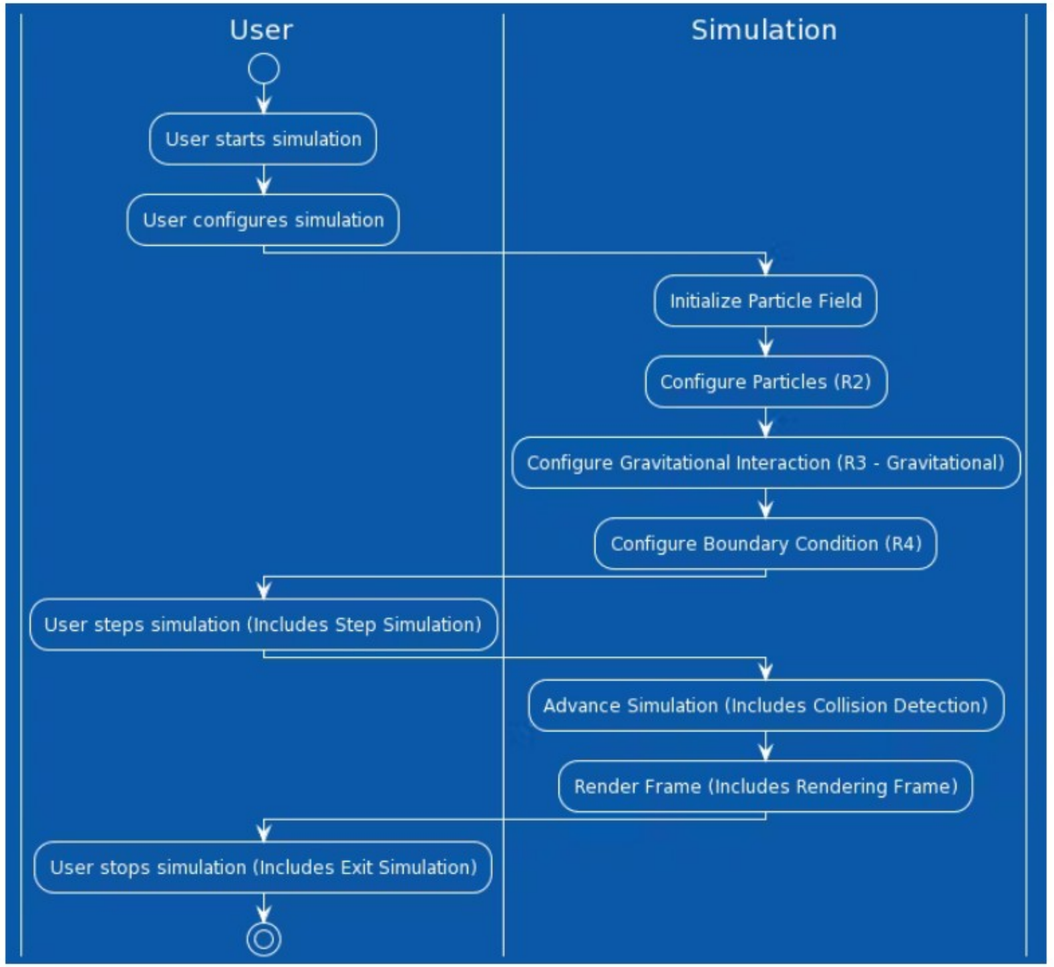
Use Case Diagram

UPPSALA UNIVERSITET

Student Project 2

Make a usecase model for the design of the Student project

Activity Diagram



Example project activity diagram start simulation

UPPSALA
UNIVERSITET

Student Project 3

Design a sequence diagram that illustrates the flow of one of your usecases , ways that the software run through a simulation

Roles in Development

- Business Analyst – the translator
- Product Owner – the ultimate decision-maker
- Developer – the one who does the actual job
- Quality Assurance Engineer – the nitpicker
- User Experience Designer – the mind-reader
- User Interface Designer – the people-pleaser
- Software Architect – the wizard
- https://brainhub.eu/library/crucial-roles-in-software-development-team

UPPSALA UNIVERSITET

What other roles do you see surounding a Development Process

Waiting for responses ...

The Role of Lead

in alarger scale project with many team members some one is assigne, given or takes the role of the Lead, the job of the Lead is to take responsibility for and to drive the part that they have responisbility for.

UPPSALA UNIVERSITET

Object

"What we mean by an object is an entity able to sabe its state(information) and which offers a number of operations(behaviour) to either examine or affect the state"

-Ivar Jacobson et al., Object Oriented Software Engineering A Use Case Driven Approach 1992(ISBN 0-201-54435-0)
https://www.gettextbooks.com/author/Ivar_Jacobson

Obejct oriented development

Object orientation is primarily a design paradigm unlike Imperative,procedural,functional and declarative programming which focuses on how you code. One can use any number of programming paradigms to implement the object-oriented design.

An object is the representation of a thing or concept, that encapsulates both data and the actions performed on it. A key concept of an object is that it interacts with the world through message passing of its parameters.

UPPSALA UNIVERSITET

Object orienteted Programming Paradigm

"Programming Paradigms
Object-oriented programming is a technique for
programming – a paradigm for writing ‘‘good’’ pro-
grams for a set of problems. If the term ‘‘object-oriented
programming language’’ means anything it must
mean a programming language that provides mechanisms
that support the object-oriented style of program-
ming well.
There is an important distinction here. A language is said
to support a style of programming if it pro-
vides facilities that makes it convenient (reasonably easy,
safe, and efficient) to use that style. A language
does not support a technique if it takes exceptional effort
or exceptional skill to write such programs; it
merely enables the technique to be used. "
- Bjarne Stroustrup
https://www.stroustrup.com/whatis.pdf

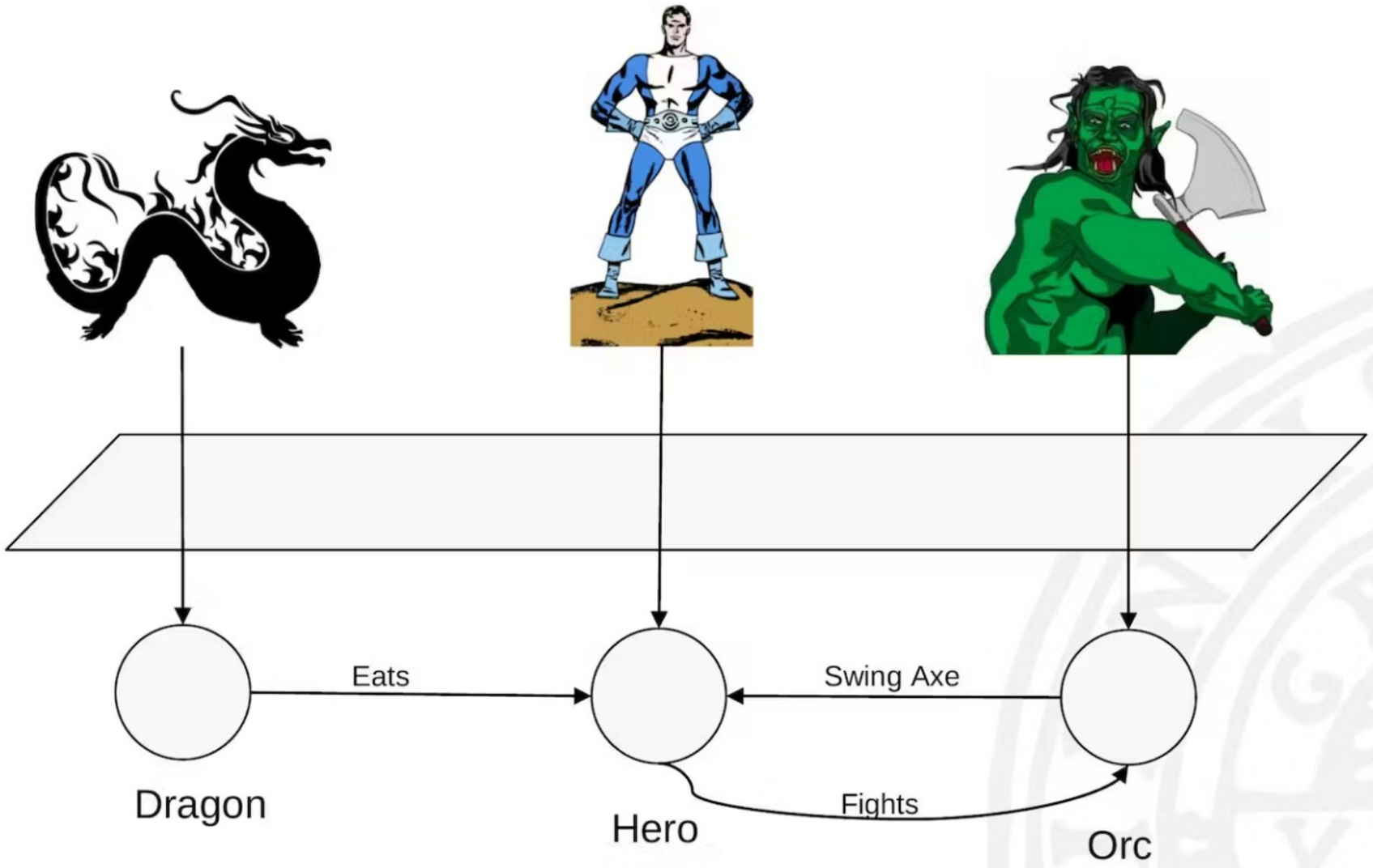
UPPSALA UNIVERSITET

Objects And obejectorientation

Dragon

Hero

Orc

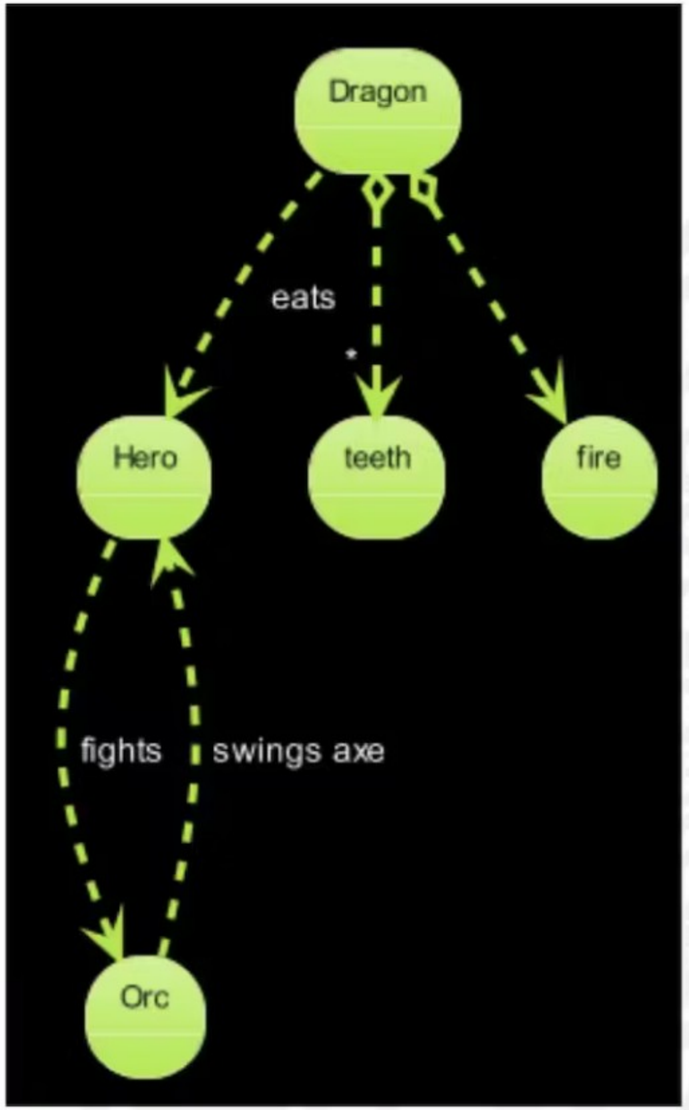


UPPSALA UNIVERSITET

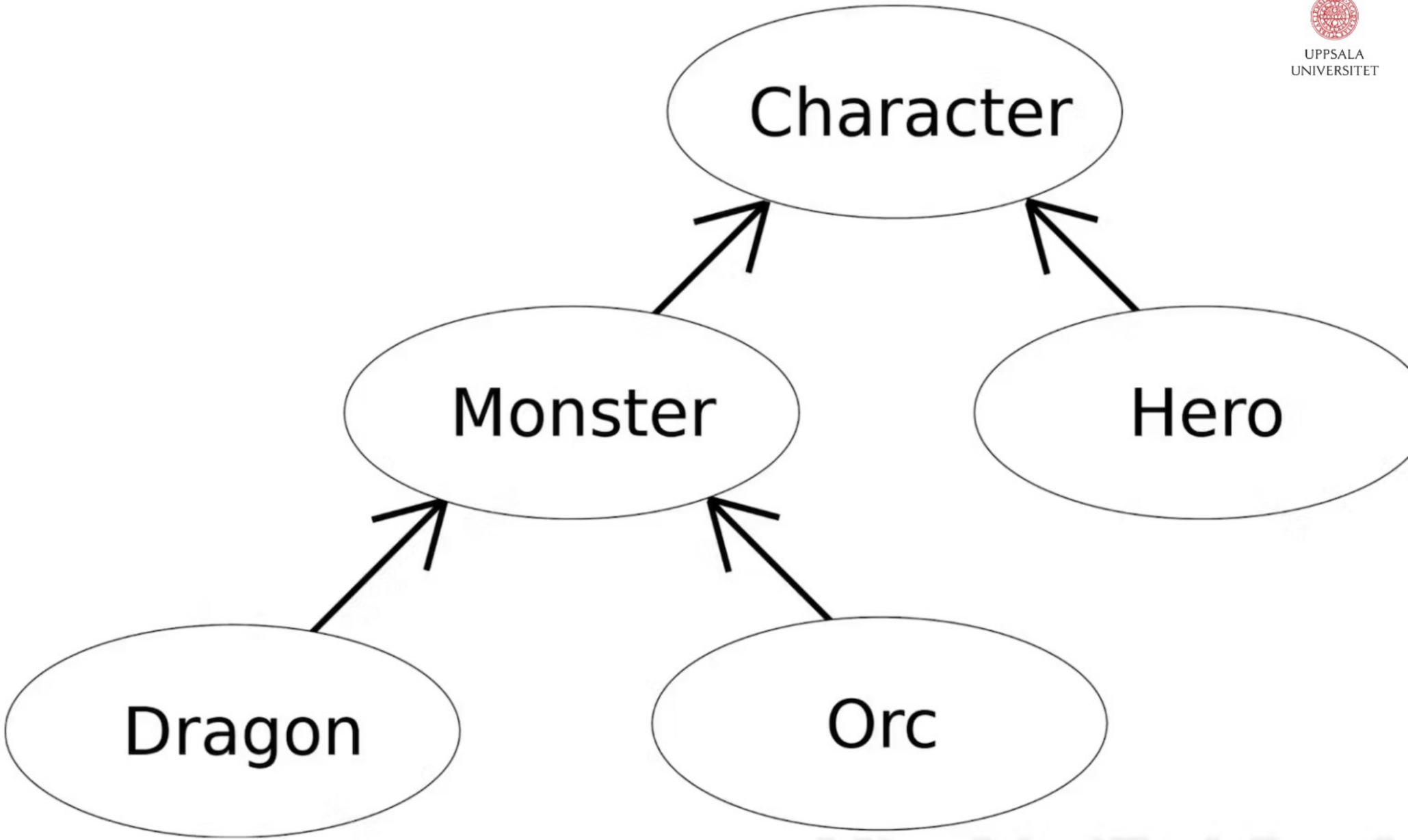
```
@startuml
!theme hacker
object Dragon
object Hero
object Orc
object teeth
object fire

Dragon..>Hero:eats
Hero..>Orc:fights
Orc..>Hero:swings axe
Dragon o..> “*” teeth
Dragon o..> fire

@enduml
```



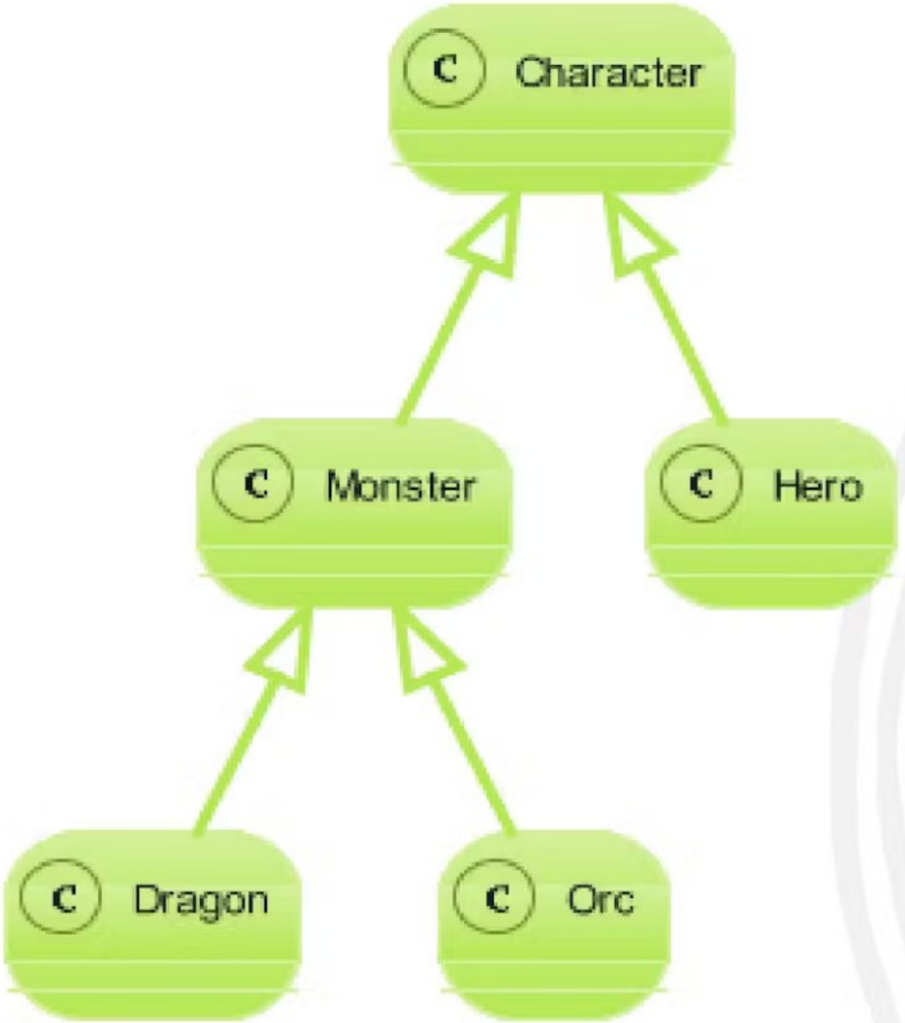
UPPSALA UNIVERSITET

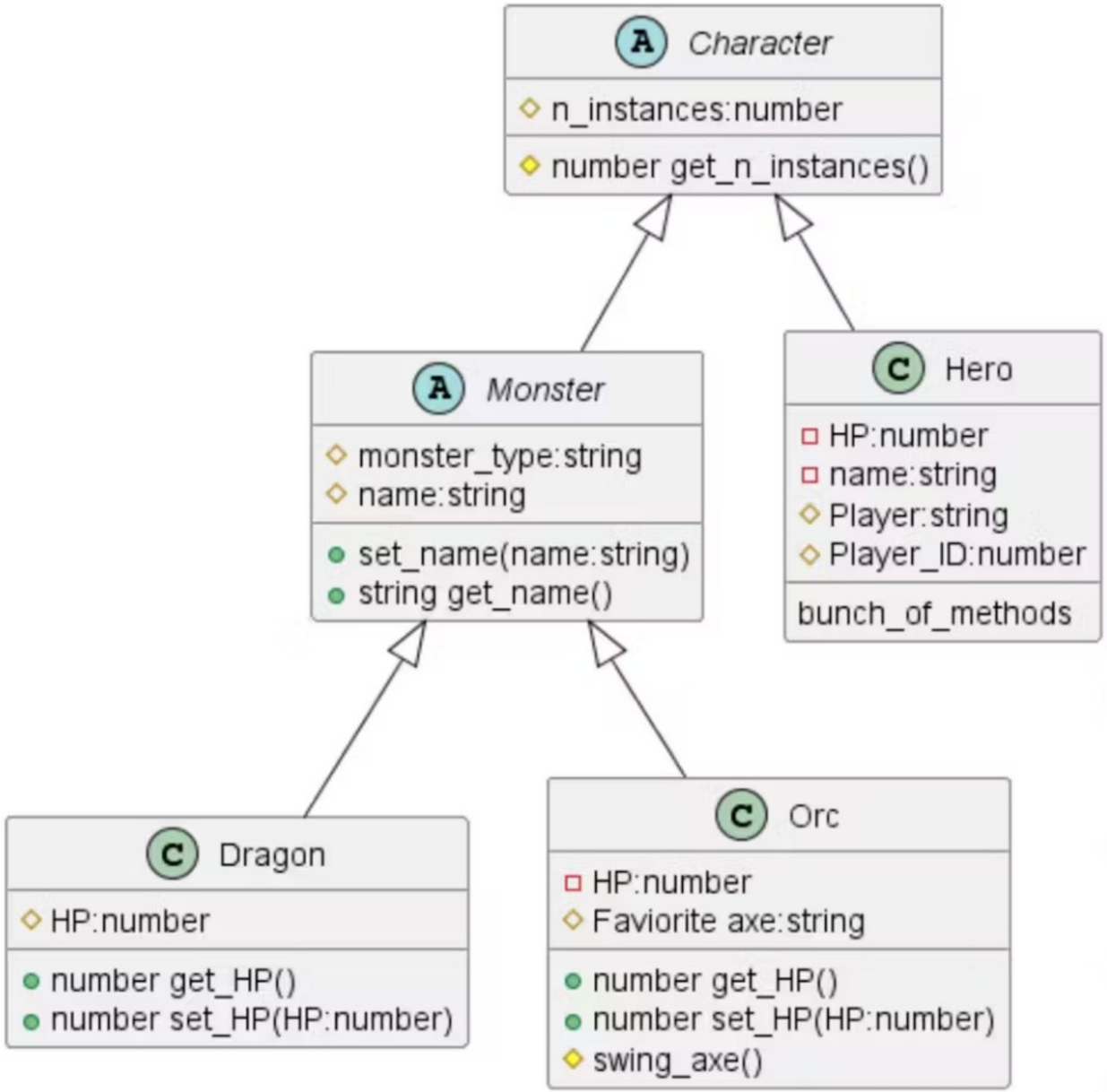


UPPSALA
UNIVERSITET

UPPSALA UNIVERSITET

Class Diagram





UPPSALA UNIVERSITET

Escersise

Make a class diagram of your current structure of classes and check what is missing ad these ass a new issue.

Modular programming

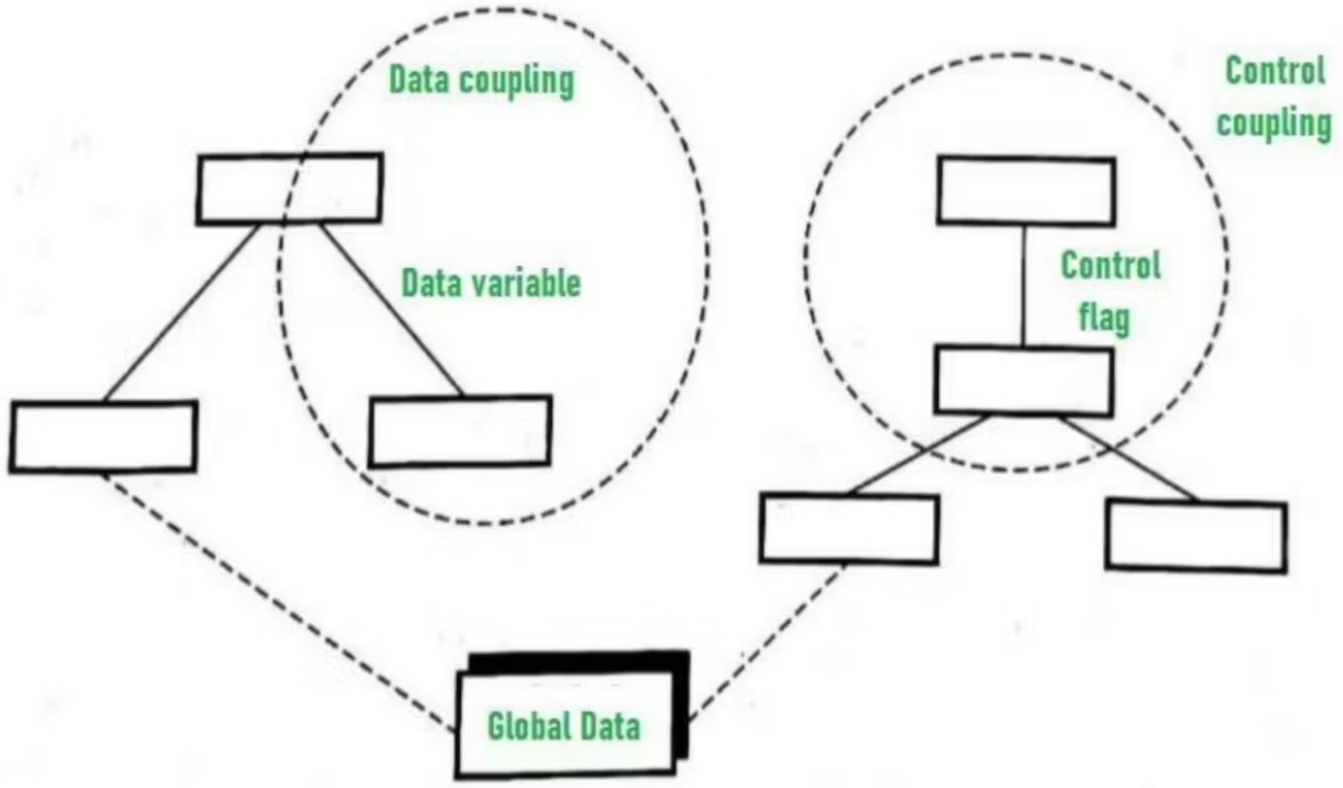
Program Control Module: In this category, a program is controlled by an independent module uniquely designed for this purpose only. The other programs may use the identical module with a similar name, but depending on the program, the module's content is designed variably.

Specific Task Module: In this category, a module is produced to achieve a particular task that is prevalent in several programs. Specific task modules are previously coded and examined, so it is easy to trust them to compose an extensive program efficiently. However, noticing this functionality of specific task modules, we also refer to them as foundational elements

Modules carries the functionality of the program and have a set of predefined data transmission options:

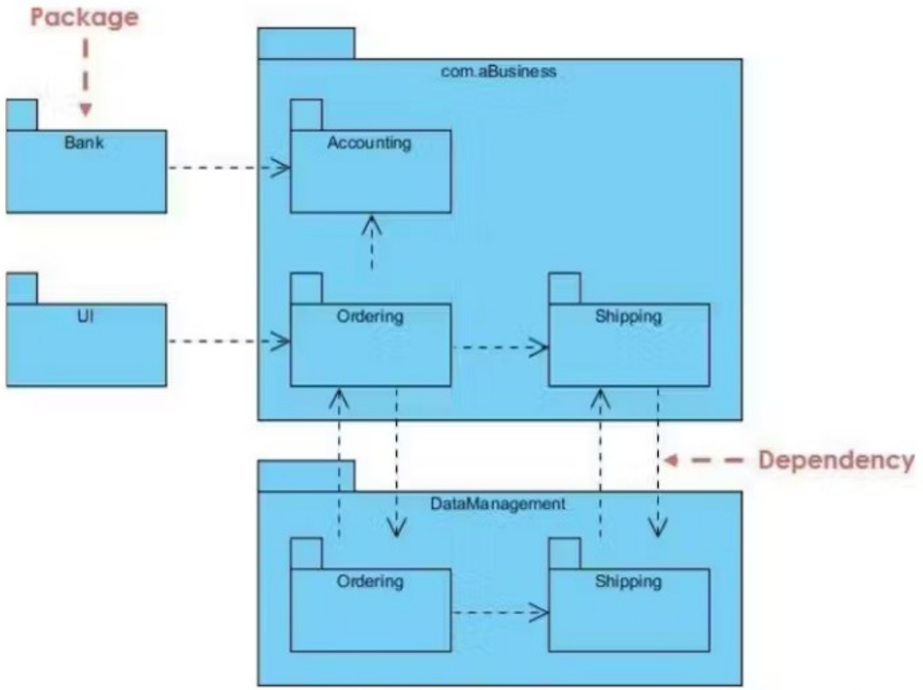
- no communication in with no communication out
- no communication in with some communication out
- some communication in with some communication out
- some communication in with no communication out

Modular programming



UPPSALA UNIVERSITET

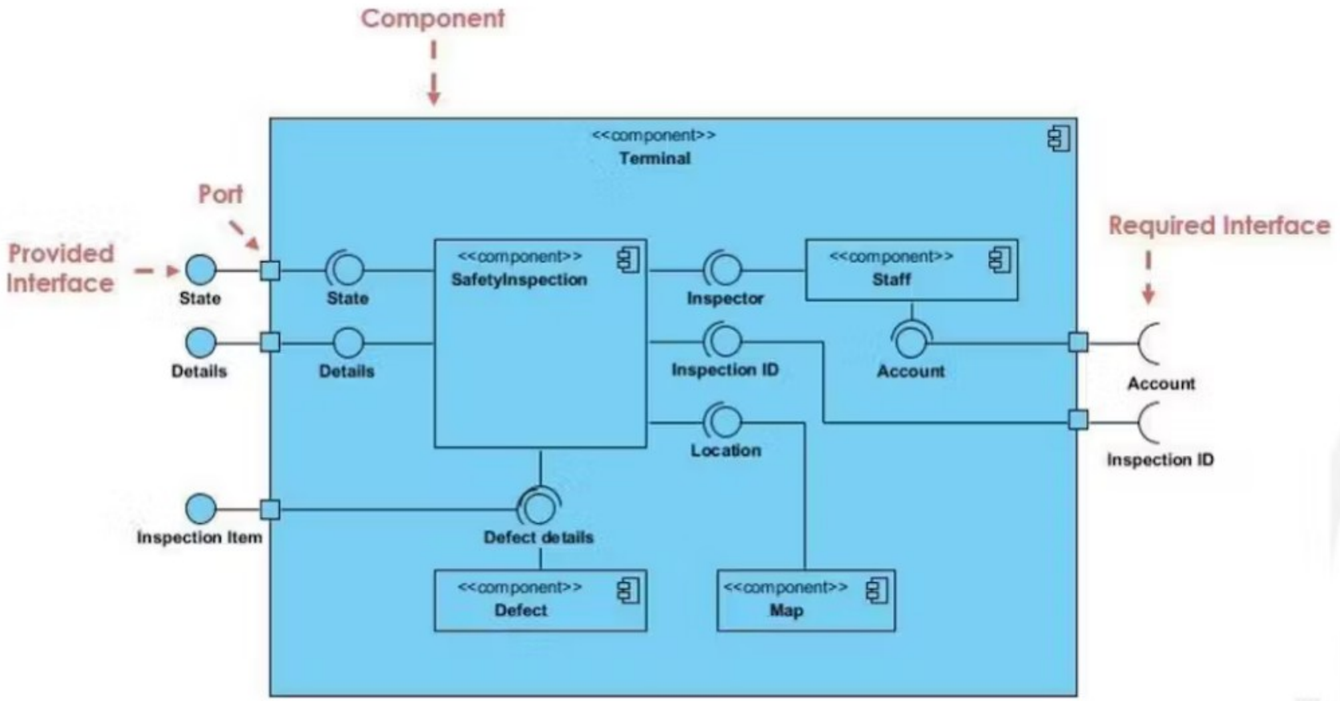
Package Diagrams



https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/

UPPSALA UNIVERSITET

Component diagrams show the deployment structure of your system



https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

Escercise Redesign for modularity

Redesign your class structure to fullfill the requirements of modularity Each module should be a logical container for the collection of classes/submodules that it contains. These documents should be first developed using issues and branches of issues then merged by the lead devloper to the develop branch and then upon merge request from the lead developer the lead deployment engineer takes responsibility to merge the develop version taged for realease. These design documents should be on github as well.

Packaging your module for others to use

In python the common package repository is PyPI Python Package Index accessed through PiP

https://packaging.python.org/en/latest/tutorials/packaging-projects/

In java the common way to distribute your module or as it is known i java package is in a jar file.

https://docs.oracle.com/javase/tutorial/deployment/jar/

In R packaging and submitting your R code to CRAN is the best way to distribute your package

https://cran.r-project.org/ and your package needs to comply with the repository guidelines found on the webpage

To help you create a package this book https://r-pkgs.org/ will guide you through the process.

C and c++ library creation is a bit more complex as you have static and dynamic linked libraries one such tool is https://cmake.org/cmake/help/latest/guide/tutorial/index.html

Fortran also has a more complex packaging and library structure both of the above have multitude of tools and ways of compiling libraries https://fortranwiki.org/fortran/show/Build+tools.

Escersie Refactor your code

Recfactor your code in such away that you create logical structural packages with clearly defined interfaces as you created in the component diagrams

Design Patterns

a **software design pattern** is a general, reusable solution to a commonly occurring problem within a given context in software design. It is not a finished design that can be transformed directly into source or machine code. Rather, it is a description or template for how to solve a problem that can be used in many different situations. Design patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

- Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. ISBN 978-0-201-63361-0.
- Brinch Hansen, Per (1995). *Studies in Computational Science: Parallel Programming Paradigms*. Prentice Hall. ISBN 978-0-13-439324-7.
- Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter (1996). *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons. ISBN 978-0-471-95869-7.
- Beck, Kent (1997). *Smalltalk Best Practice Patterns*. Prentice Hall. ISBN 978-0134769042.
- Schmidt, Douglas C.; Stal, Michael; Rohnert, Hans; Buschmann, Frank (2000). *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. John Wiley & Sons. ISBN 978-0-471-60695-6.
- Fowler, Martin (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley. ISBN 978-0-321-12742-6.
- Hohpe, Gregor; Woolf, Bobby (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley. ISBN 978-0-321-20068-6.
- Freeman, Eric T.; Robson, Elisabeth; Bates, Bert; Sierra, Kathy (2004). *Head First Design Patterns*. O'Reilly Media. ISBN 978-0-596-00712-6.

UPPSALA
UNIVERSITET

Thank you and happy coding in the optimisation part