STI · INNSBRUCK

OO modeling

MSc 2008/2009

Lecture 7/8

Summary of the previous lecture

STI · INNSBRUCK

- ER modeling
  - General principles.
  - Basics: entity types, relationships, cardinality, existence, entity subtypes, attributes.
  - Advanced notions: degrees of a relationship, relationship attributes, partitions, aggregations, weak entities etc.

Today‘s lecture

STI · INNSBRUCK

- Object-oriented modeling.

The structured analysis

STI · INNSBRUCK

- The functional view defines what is to be done and the data flows between the things to be done.
  - Is made up of data flows diagrams.
- The data view captures the static structure of the system. It describes what is in or outside the system.
  - Is usually made up of entity relationship diagrams.
- The dynamic view describes when things happen and under which circumstances.
  - Is usually made up of state transition diagrams.

The object-oriented analysis

STI · INNSBRUCK

- OOA models requirements in terms of objects and services they provide.
- The object model (data view) describes the things in or outside the system, and their relationships.
- Interactions are mapped into this object model.
- Difference to structural analysis
  - Processes change more than objects themselves. Structural analysis: changes in functionality result in changes in the software structure.

The object-oriented model

STI · INNSBRUCK

- The model consists of a collection of objects.
- An object contains values stored in instance variables within the object.
- An object also contains methods, which are code that can be executed on the object. An object A can access the data of another object B by invoking the method of object B (message passing).
- An object has an internal part of attributes and methods that is not visible from the outside.
- Objects have unique identity.

Differences to the ER model

STI · INNSBRUCK

- Both are made up of a collection of objects/entities.
- OO objects may contain other objects (nesting).
- OO objects contain methods.
- Both object collections are organized into hierarchies.
- Unlike ER entities, each object has its own unique identity:
  - Two objects containing the same values are distinct.
  - Distinction is ensured a the physical level.

Modeling primitives

STI · INNSBRUCK

- Objects
  - Entity with state, attributes and services.
- Classes
  - Collections of similar entities organized in specialization/generalization hierarchies.
- Attributes
  - Together represent the state of an object.
  - Types, visibility, modifiability.

Modeling primitives (cont)

STI · INNSBRUCK

- Relationships
  - Associations, aggregations, etc. between objects.
  - Specialization/generalization between classes.
- Methods (services, functions)
  - Operations that all objects of a class can execute when called by other objects (message passing).
- Use cases
  - Sequences of message passing between objects representing interactions.

General principles

STI · INNSBRUCK

- Inheritance
  - Subclasses inherit attributes and methods from superclasses (multiple inheritance).
  - Subclasses specialize the superclass by adding new attributes, methods or modifying existing ones.
- Information hidding
  - Internal state of an object is hidden from the outside world.
  - Objects may contain other objects and hide their services (abstraction).

UML

STI · INNSBRUCK

- UML stays for Unified Modeling Language
  - Originally developed at Rational Software
  - Managed currently by the Object Management Group (see link).
- UML is a modeling language (syntax).
- UML is primarily aimed at building models of software. It can but can be used for any modeling task.

- UML provides a number of diagram types that can be used to capture knowledge in terms of specific, interconnected model elements.

Characteristics of UML

STI · INNSBRUCK

- Syntax.
  - UML specifies what model elements and diagrams are available and the rules associated with them.
  - UML does not specify what diagrams to create and how.
- Application independence.
  - UML can be used to model everything and can be extended to accommodate user requirements.
- Programming language independence.
  - UML models can be mapped into code based on a case tool.

Characteristics of UML (cont)

STI · INNSBRUCK

- Process independence.
  - UML does not specify the way models should be built.
- Tool independence.
  - A wide range of tools for visualizing UML models are available. UML as a language is tool-independent.

Diagram types (UML 2.0)

STI · INNSBRUCK

- **Structural modeling diagrams** define the static architecture of a model. They are used to model the things that make up a model: classes, objects, interfaces and physical components. In addition, they are used to model the relationships and dependencies between elements.
  - **Package diagrams** divide the model into logical packages and describe the interactions between them.
  - **Class or structural diagrams** define the structural elements of a model: the types, classes, attributes, methods etc.
  - **Object diagrams** show how instances of structural elements – objects - are related and used at run-time.
  - **Composite structure diagrams.**
  - **Component diagrams.**
  - **Deployment diagrams.**

http://www.sparxsystems.com/resources/uml2_tutorial/index.html

Diagram types (UML 2.0) (cont)

STI · INNSBRUCK

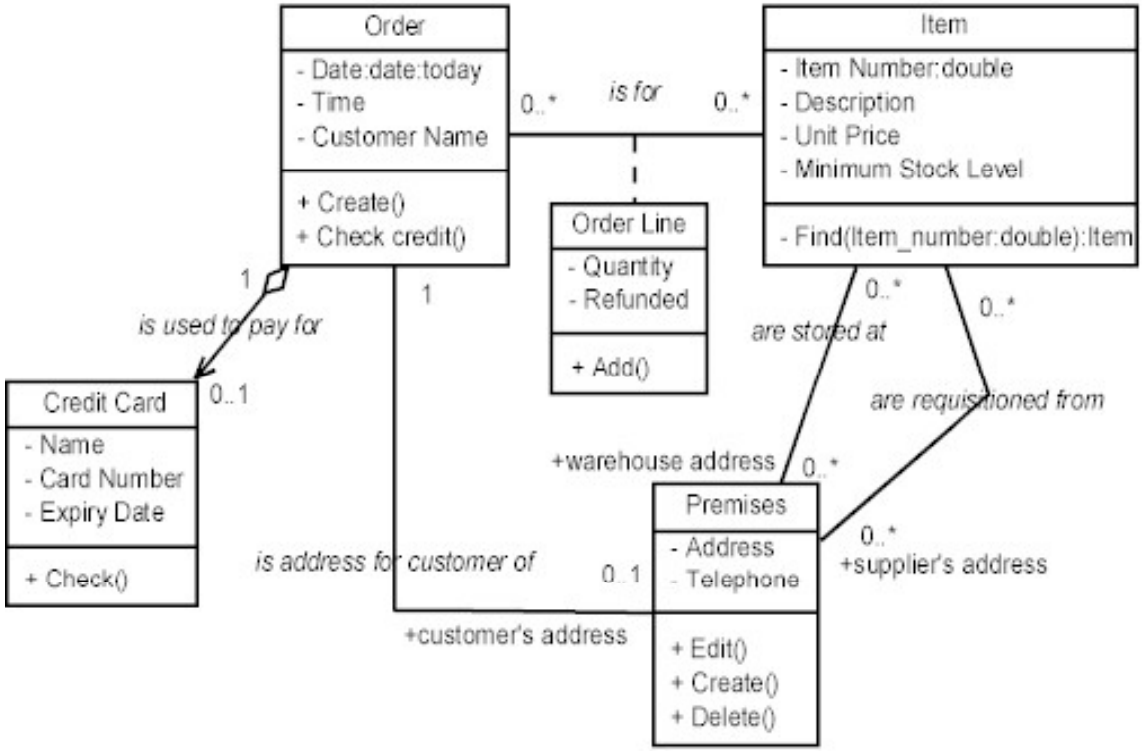
- **Behavioral Modeling Diagrams** capture the interactions and the states within a model that runs over time.
  - **Use case diagrams** model the interaction between users and system.
  - **Activity diagrams** can be used to define the general program flow, and to capture the decision points and actions within any generalized process.
  - **State machine diagrams** capture the "run state" of a model when it executes.
  - **Communication** diagrams.
  - **Sequence diagrams** show the sequence of messages passed between objects using at a point in time.
  - **Timing diagrams**.
  - **Interaction overview diagrams**.

Class diagrams

STI · INNSBRUCK

- Class diagrams show the static structure of the systems.
- The class diagram shows the building blocks of any object-orientated system. Class diagrams depict a static view of the model, or part of the model, describing what attributes and behavior it has rather than detailing the methods for achieving operations. Class diagrams are most useful in illustrating relationships between classes and interfaces. Generalizations, aggregations, and associations are all valuable in reflecting inheritance, composition or usage, and connections respectively.
-

Example



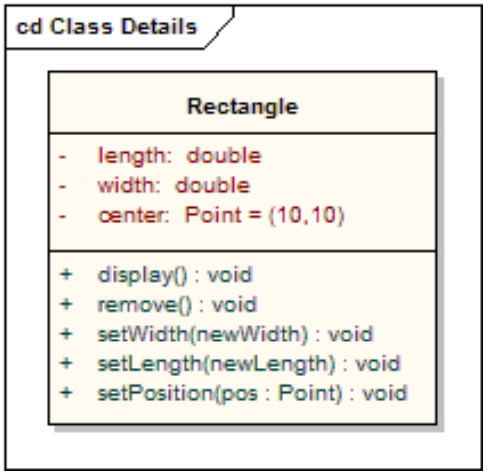
The symbol that precedes the attribute, or method name, indicates the visibility of the element:

+ means public .
- means private.
means protected.
~ means package-level visibility.

Classes

STI · INNSBRUCK

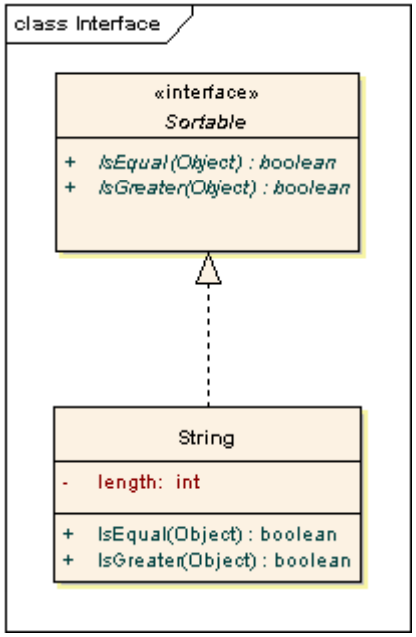
- Classes are collections of objects sharing the same attributes and methods.
- Attributes capture the data properties of the classes including type, default value and constraints
- Methods capture the signature of the functionality - parameters, parameter types, parameter constraints, return types and the semantics.



Interfaces

STI · INNSBRUCK

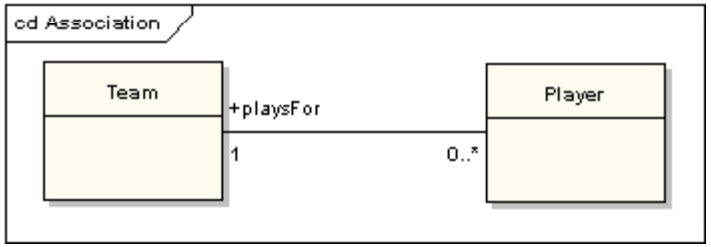
- An interface specifies a certain behavior that developers agree to meet.
- Classes realize interfaces.

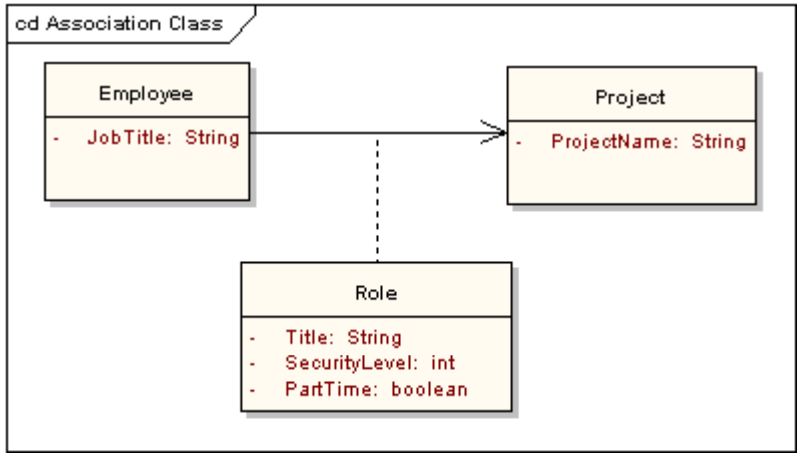


Associations

STI · INNSBRUCK

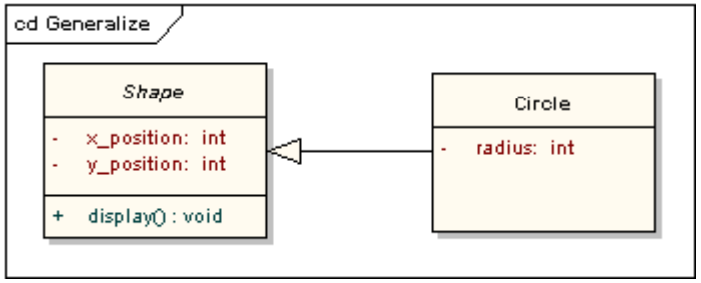
- Associations describe interactions between objects of different classes.
- They can have further properties, and cardinalities.



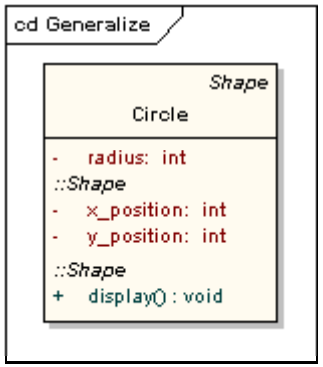


Generalizations

- A generalization indicates inheritance between classes.



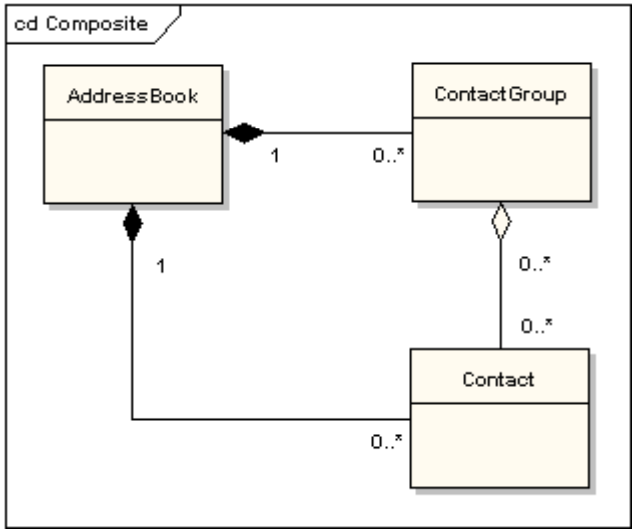
**The class "Shape" is abstract, shown by the name being italicized.*



Aggregations

STI · INNSBRUCK

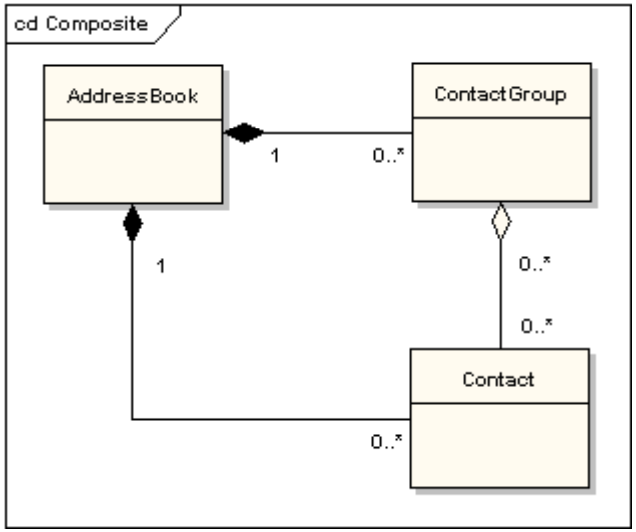
- Aggregations indicate elements that are made of other components.
- A composite aggregation -shown by a black diamond –is used when components can be included **in a maximum of one** aggregation at a time.
- The deletion of the composite element implies the deletion of the parts.
- A part can be individually removed from a composition without having to delete the entire composition.



Weak vs. strong aggregations

STI · INNSBRUCK

- An address book is made up of a contacts and contact groups.
- A contact group virtually groups contacts.
- A contact belongs to zero or more contact groups.
- Deleting an address book means deleting all contacts and contact groups.
- Deleting a contact group does not imply deleting contacts.



Object diagrams

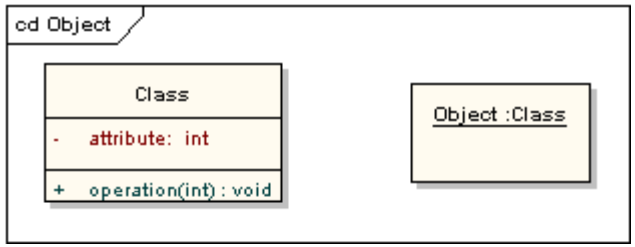
STI · INNSBRUCK

- Object diagrams how objects are related and used at run-time.
- They can be considered a special case of class diagrams.
- They emphasize the relationship between objects at some point in time.

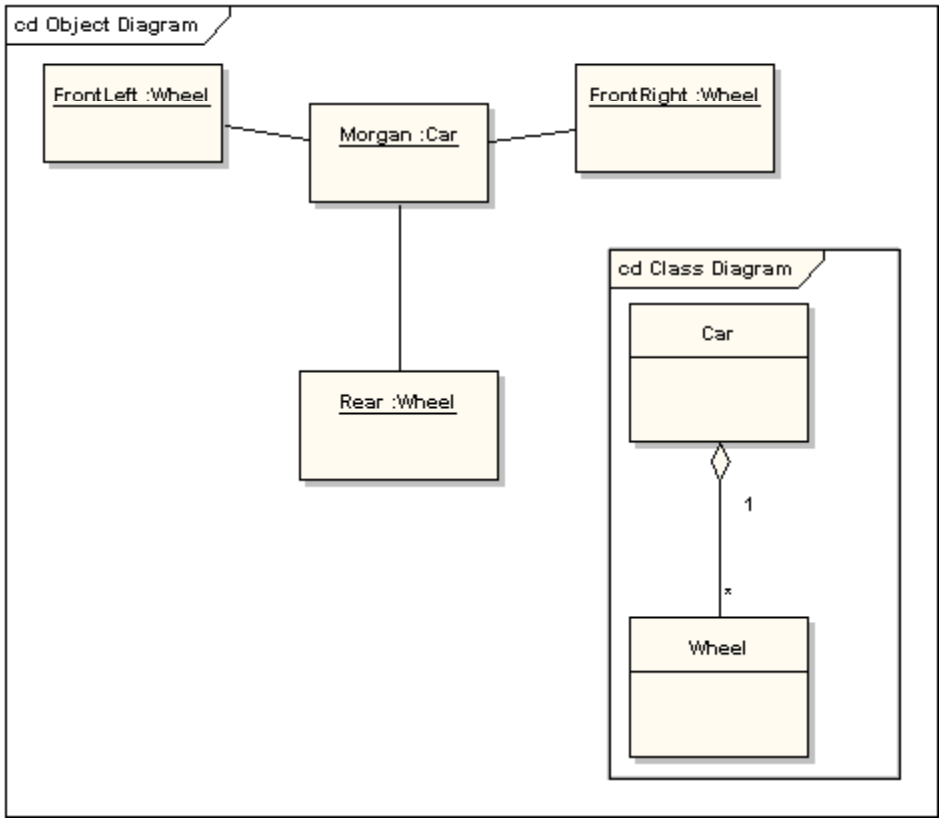
Objects

STI · INNSBRUCK

- Object elements do not depict attributes and methods.
- The display of names is different: object names are underlined and may show the name of the class.

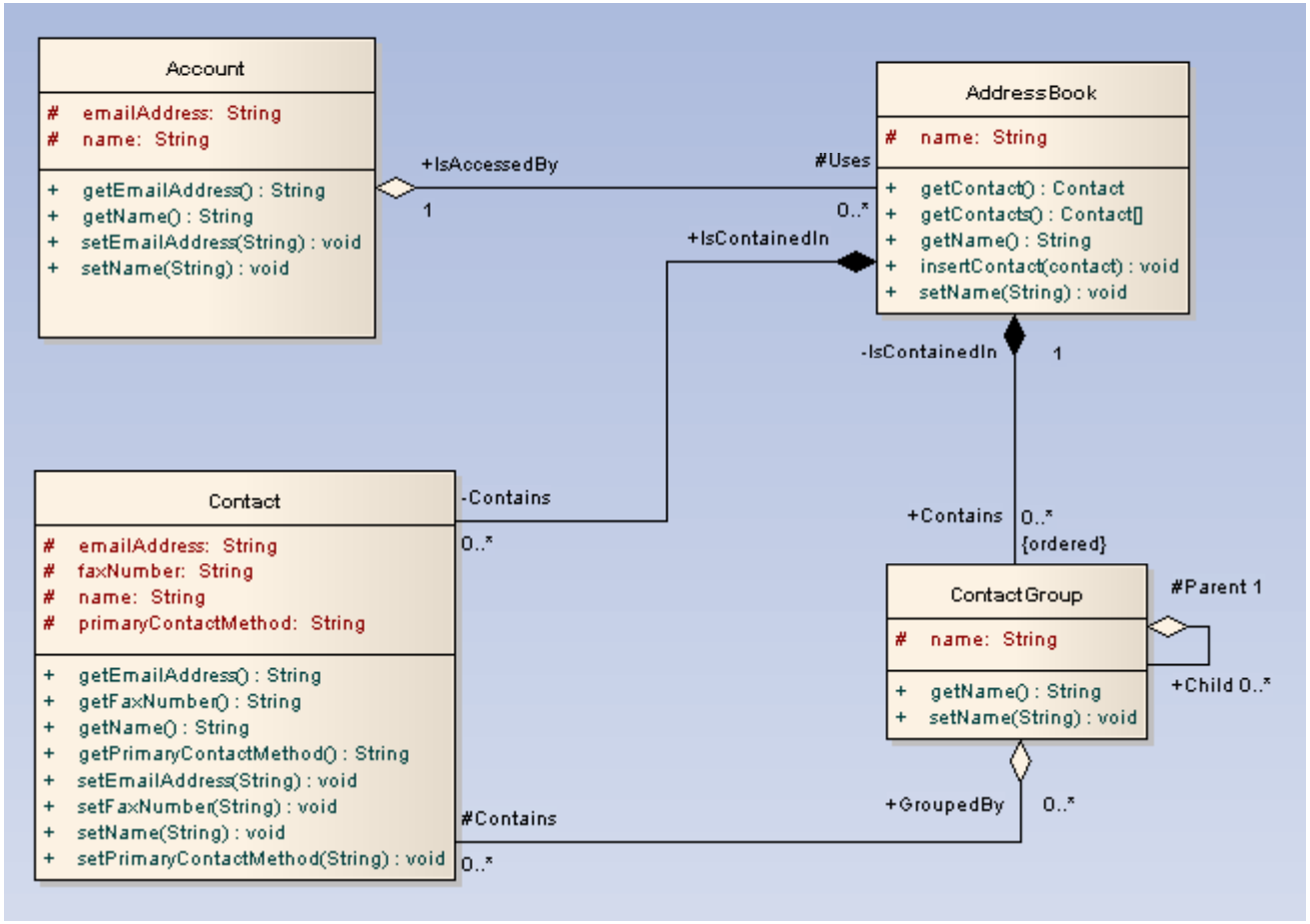


Class and object diagrams



Example

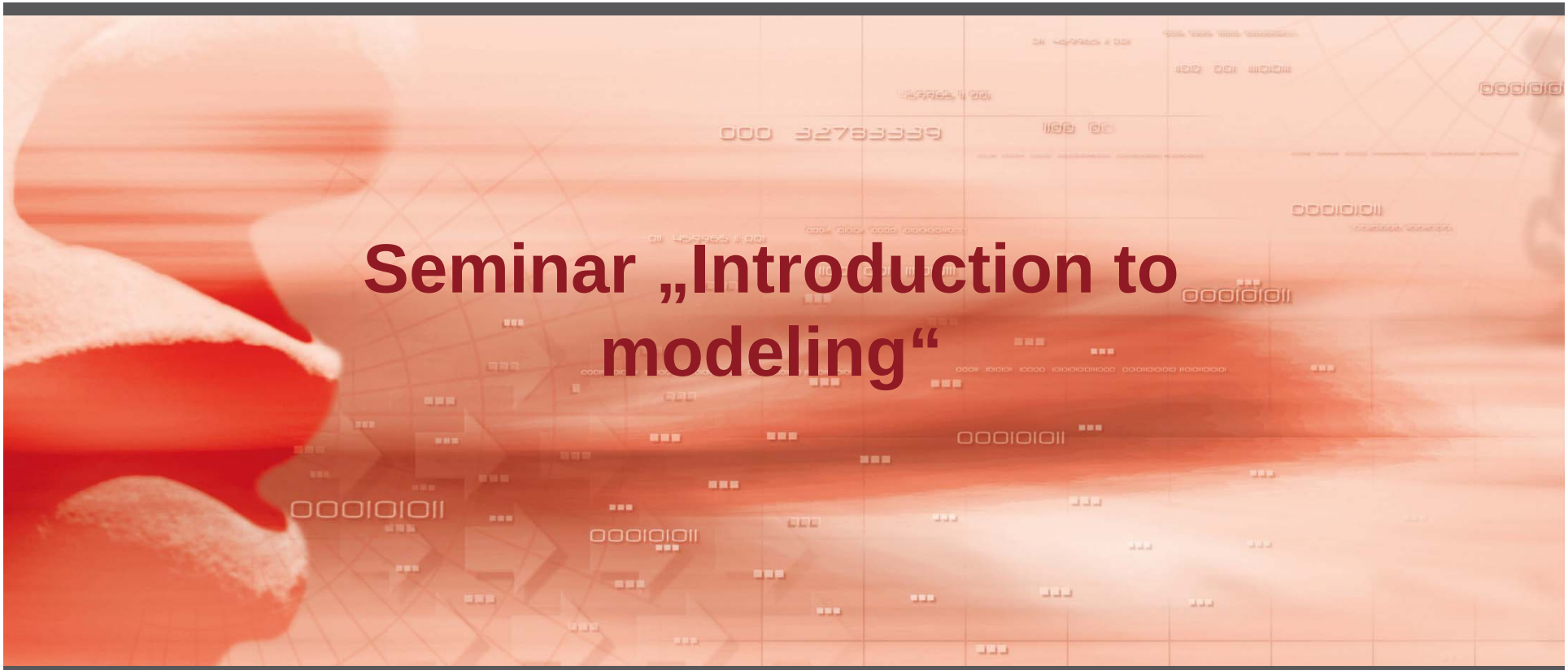
STI · INNSBRUCK



Summary

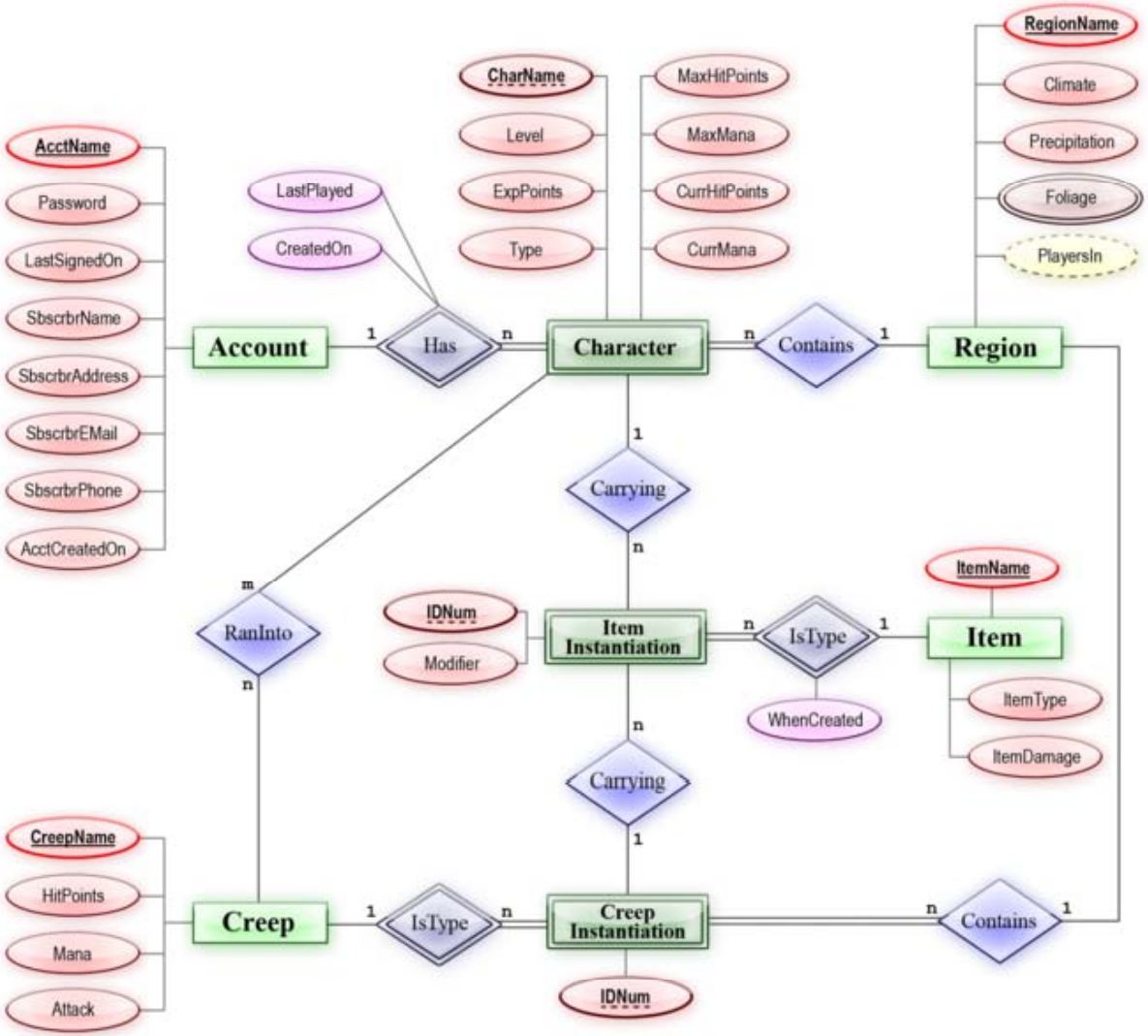
STI · INNSBRUCK

- OO modeling.
- UML.
- Class and object diagrams.
- Relationships to ER modeling.



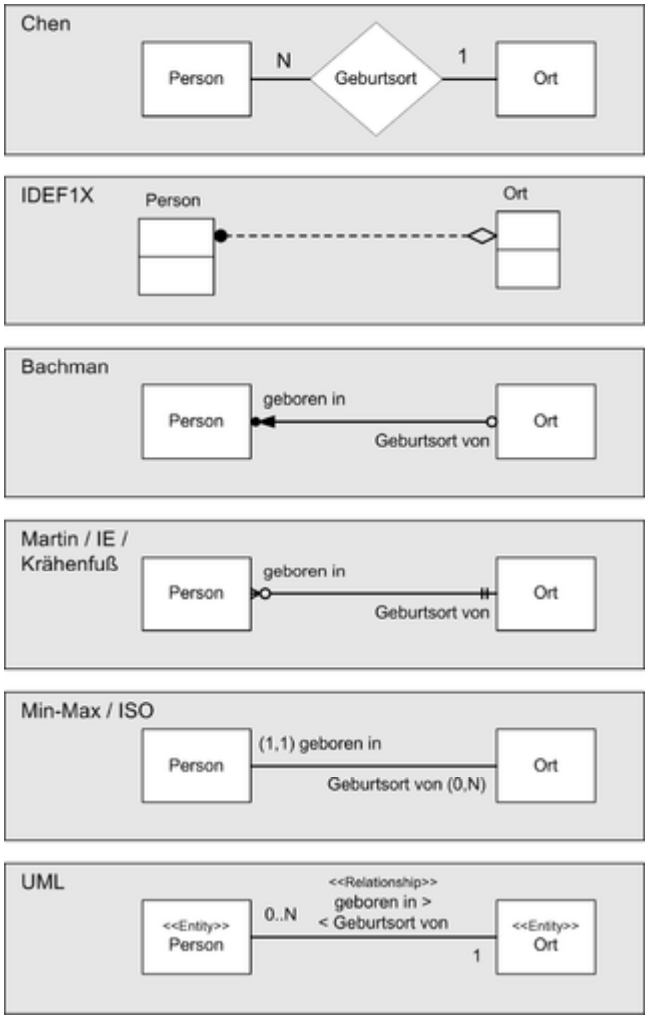
Seminar „Introduction to modeling“

Notations



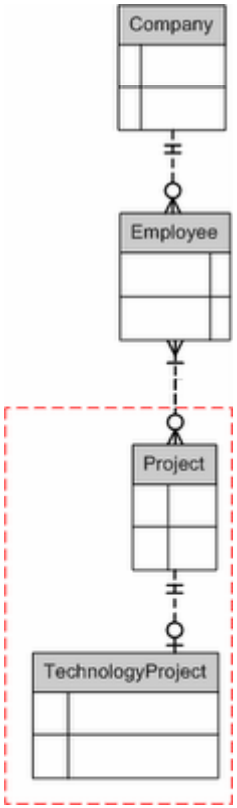
http://en.wikipedia.org/wiki/Image:ER_Diagram_MMORPG.png

Notations (cont)



http://en.wikipedia.org/wiki/Entity-relationship_model

Crow‘s Foot notation



http://rapidapplicationdevelopment.blogspot.com/2007/06/entity-relationship-diagram-example.html

Assignments 7/8

STI · INNSBRUCK

a. Introduce the most popular ER notations and the differences between them.

b. Give an ER design to model the following scenario. Use the Crow's Foot notation (Assignment based on http://www.vocw.vn/content/m10489/latest/)

1. Patients are identified by SSN, name, address and age.
2. Doctors are identified by SSN. For each doctor, the name, specialty and years of experience must be recorded.
3. Each pharmacy has a name, address and phone number. A pharmacy must have a manager.
4. A pharmacist is identified by an SSN, he/she can only work for one pharmacy. For each pharmacist, the name, qualification must be recorded.
5. For each drug, the trade name and formula must be recorded.
6. Every patient has a primary physician. Every doctor has at least one patient.
7. Each pharmacy sells several drugs, and has a price for each. A drug could be sold at several pharmacies, and the price could vary between pharmacies.
8. Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and quantity associated with it.
9. State all assumptions used in developing your data model.

Assignments 7/8 (cont)

STI · INNSBRUCK

c. Draw the corresponding ER diagram for the following scenario. Use the UML notation.

1. Each supplier has a unique name.
2. More than one supplier can be located in the same city.
3. Each part has a unique part number.
4. Each part has a color.
5. A supplier can supply more than one part.
6. A part can be supplied by more than one supplier.
7. A supplier can supply a fixed quantity of each part.

Thank You!

Questions?