Software Engineering

Class Object Diagrams

Dr. Sayed AbdelGaber

Professor

Faculty of Computers and Information

Helwan University

OBJECTS AND OBJECT CLASSES

- Objects are entities in a software system which represent instances of real-world and system entities.
- Object classes are templates for objects. They may be used to create objects.
- Object classes may inherit attributes and services from other object classes.

OBJECT COMMUNICATION

- Conceptually, objects communicate by message passing.

- Messages
  - ✓ The name of the service requested by the calling object;
  - ✓ Copies of the information required to execute the service and the name of a holder for the result of the service.

- In practice, messages are often implemented by procedure calls
  - ✓ Name = procedure name;
  - ✓ Information = parameter list.

WHAT ARE CLASS DIAGRAMS?

- **A diagramming technique that documents the static, structural aspects of an object-oriented system:**
  - ✓ **Types of object classes to be stored in the system and the properties associated with each object class.**
  - ✓ **Relationships (associations) among these object classes.**
  - ✓ **Behavior associated with each class.**

- **Core diagram as they represent 'building blocks of any object-oriented system'.**

[Source: Bennet et al., 2001]

- **Describes the 'abstract not the concrete':**
  - ✓ **Object instances modeled in Object Diagram.**

PURPOSE

- Document classes that constitute a system or subsystem.
- Show individual features of each class.
- Used throughout development process:

  From: specification classes (requirements) in problem domain.

  To: implementation model of proposed system.

- Describe associations, generalization, aggregation relationships between classes.

[Source: Bennet et al., 2001]

CLASSIFICATIONS OF CLASSES

- **Entity:**
  - ✓ Model information requirements and associated relationships.
  - ✓ Could be a person role, tangible object, event, etc.
  - ✓ Will form database structure → persistent.
- **Boundary:**
  - ✓ Model interactions between a system and it's actors.
  - ✓ Represents classes for user interface.
- **Control:**
  - ✓ Controls other objects.
  - ✓ Represents classes for processing.
- **Can package and identify class types:**
  - ✓ Package diagrams
  - ✓ Stereotypes

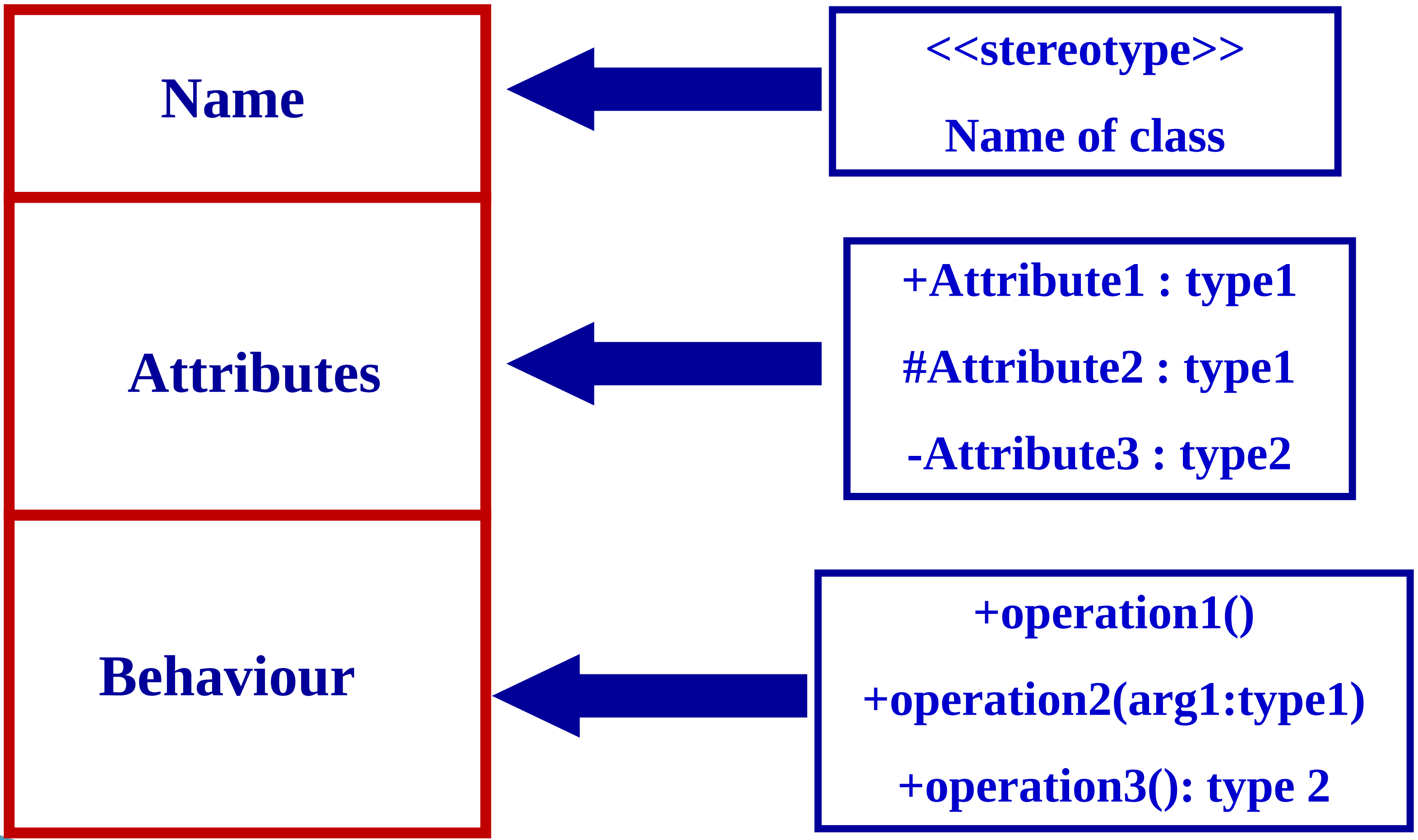
COMPONENTS OF CLASSES

- **Classes are denoted by rectangles divided into 3 parts:**
  - ✓ **Name → unique text description**
  - ✓ **Attributes → name + data type**
  - ✓ **Behavior → operation signatures**

- **Can add further meaning using stereotypes, visibility, and by grouping related operations.**

- **Notice no relationship properties:**
  - ✓ **Represented as associations.**
  - ✓ **Associations are a separate element of a class.**

CLASS COMPONENTS TEMPLATE



Stereotypes

- **Phrase surrounded by guillemets: <<name>>.**
- **Used to convey additional semantics or classify diagram elements:**
  - ✓ **Extension mechanism.**
  - ✓ **Reduce ambiguity.**
  - ✓ **Describe purpose.**
- **Standard stereotypes:**
  - ✓ **<<include>>, <<extend>>, <<interface>>, <<abstract>>, ...**
- **Often used stereotypes:**
  - ✓ **<<entity>>,<<subsystem>>, <<persistent>>, <<constructor>>, ...**
- **New stereotypes can be defined by analyst/designer.**

ATTRIBUTE TYPES

- **Primitive types:**
  - ✓ **Atomic.**
  - ✓ **Enumerated.**
  - ✓ **Examples: String, int, character, Boolean ...**

- **Class types:**
  - ✓ **From implementation environment:**
  - ✓ **JAVA classes for Date, Integer, ...**
  - ✓ **From class model itself:**
  - ✓ **Address, Name, ...**

- **Power Designer supports generic primitive types with Design language.**

TYPES OF OPERATIONS

- **Constructor:**
  - ✓ Creates new instance of a class.
  - ✓ Can have multiple constructors → different arguments.
- **Query:**
  - ✓ Accesses the state of an object (can not modify).
  - ✓ Example: GET operation.
- **Update:**
  - ✓ Modifies state of object.
  - ✓ Example: SET operation.
- **Scope:**
  - ✓ Applies to a class (extent), not an instance.
  - ✓ Example: Aggregation of some attribute value:
    - Average commission rate for all sales people.

VISIBILITY

- Defines the availability or accessibility of an attribute or operation to other classes.

- Close relationship with data hiding and encapsulation.

- Typically:
  - ✓ attributes have *private* visibility.
  - ✓ operations have *public* visibility.

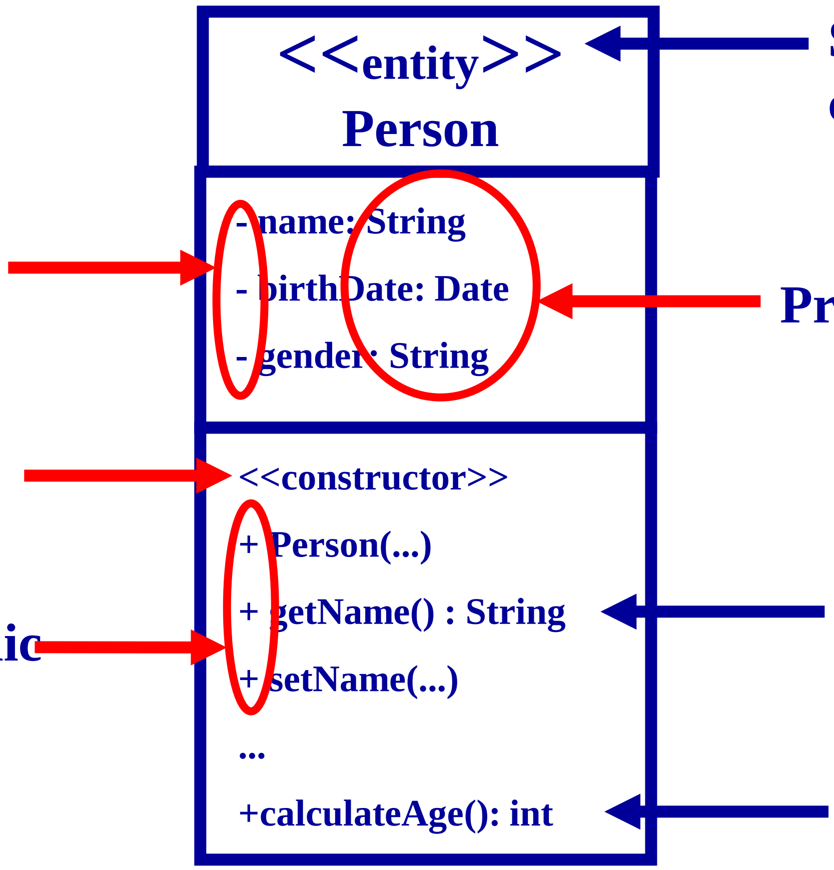
- Public operations thought of as interface for class.

See next slide

VISIBILITY OPTIONS

- **Public:**
  - ✓ **Denoted by +**
  - ✓ **Other classes may directly examine or change the feature.**
- **Protected:**
  - ✓ **Denoted by #**
  - ✓ **Only classes of a public or protected subclass (descendants) can directly examine or change the feature.**
- **Private:**
  - ✓ **Denoted by –**
  - ✓ **Only class itself (but not of inheriting classes) can directly examine or change the feature.**

Class example : PERSON



Stereotype to identify class type

Attributes with private visibility → hidden

Primitive types

Stereotype to group operations.

Operations with public visibility → encapsulated

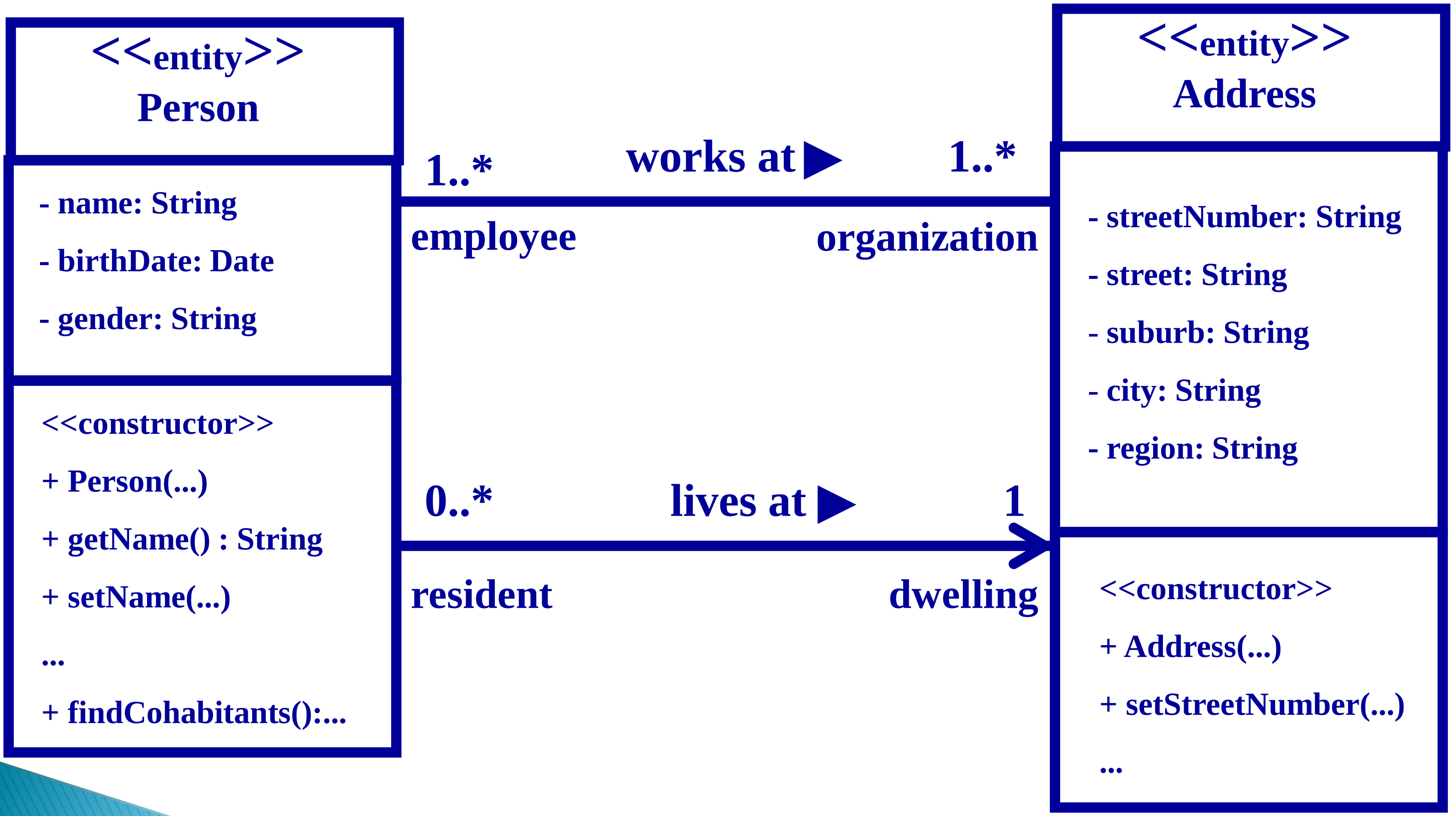
Return type

Behavior specific to each object

ASSOCIATIONS

- Represents relationship between object classes.
  - ✓ Relationship properties not modeled in class diagram.
- Denoted by solid line.
- Can be annotated with additional components:
  - ✓ Multiplicity, name, role names, navigability, qualifiers.
- Different types:
  - ✓ Composition vs. aggregation.
  - ✓ Associative classes.
  - ✓ Note not generalization.
- Associations discussed in detail

Brief association example

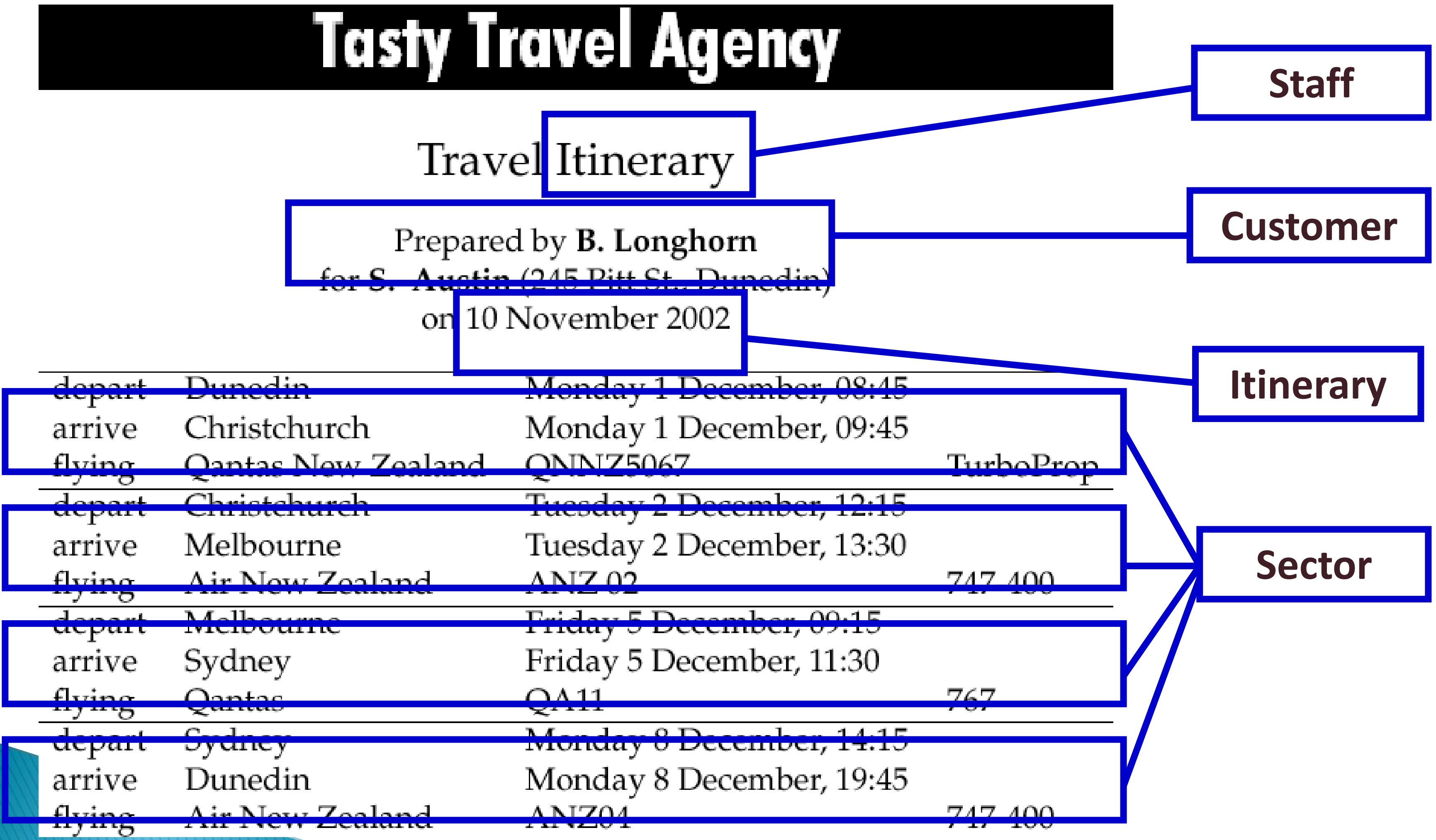


NOTE ON CONCEPTUAL MODELING

- Follow these iterative activities to produce initial, high-level class diagrams:
  1. Find classes and associations
  2. Identify attributes and operations
     - ✓ allocate to classes
  3. Identify generalization structures
- Very similar to recommended approach to ERD construction.

[Source: Bennet et al., 2001]

Recall itinerary form breakdown



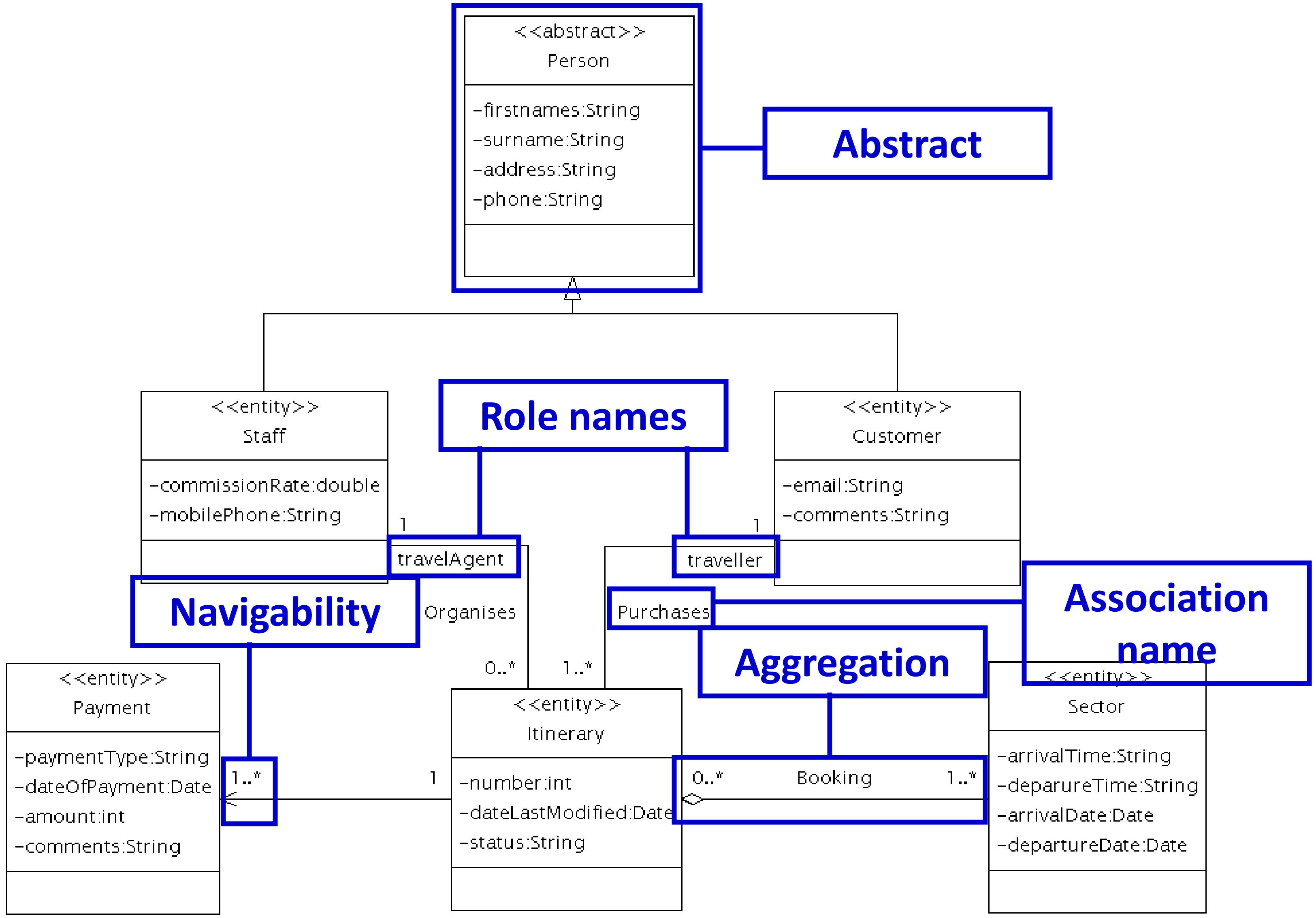
Tasty Travel Agency

Travel Itinerary

Prepared by **B. Longhorn**
for **S. Austin** (245 Pitt St, Dunedin)
on 10 November 2002

depart	Dunedin	Monday 1 December, 08:45	
arrive	Christchurch	Monday 1 December, 09:45	
flying	Qantas New Zealand	QNNZ5067	TurboProp
depart	Christchurch	Tuesday 2 December, 12:15	
arrive	Melbourne	Tuesday 2 December, 13:30	
flying	Air New Zealand	ANZ 02	747-400
depart	Melbourne	Friday 5 December, 09:15	
arrive	Sydney	Friday 5 December, 11:30	
flying	Qantas	QA11	767
depart	Sydney	Monday 8 December, 14:15	
arrive	Dunedin	Monday 8 December, 19:45	
flying	Air New Zealand	ANZ04	747-400

TASTY TRAVEL AGENCY CLASS DIAGRAM



Questions