University Application Project Report for Clark University

Names: Durvank Deorukhkar, Percival Tapera

Introduction

This report documents the design and implementation of a class hierarchy for the Clark University Application system, developed in C#. The system models various entities within the university, including students (PartTimeStudent, UnderGradStudent, GradStudent, and PhDStudent) and staff members (LabManager, Faculty, PartTimeFaculty, and Secretary). The design emphasizes object-oriented principles such as inheritance, polymorphism and encapsulation to represent shared attributes and behaviors efficiently.

A base class Person captures common attributes across all entities (e.g., name and contact information). The hierarchy branches into two primary intermediate base classes: Student for academic personnel and Employee for administrative and academic staff. Derived classes extend these bases with specialized attributes and behaviors, particularly the ComputeGrade method for students, which uses polymorphic overrides based on specific grading formulas.

For each class, two constructors are provided: a no parameter (default) constructor for initialization with default values and a parameterized constructor for direct assignment of all fields. Each class also includes a ToString method to provide a human-readable string representation of the object's state.

Class Hierarchy Overview

Base Class: Person

The Person class serves as the root of the hierarchy, encapsulating universal attributes for all university members.

Attributes:

- firstName (string): The person's first name.
- lastName (string): The person's last name.
- street (string): The street address.
- city (string): The city of residence.
- telephone (string): The contact telephone number.

Constructors:

1. Default constructor: Initializes all fields to empty strings ("").
2. Parameterized constructor: Accepts firstName, lastName, street, city, and telephone as arguments and assigns them to the respective fields.

Methods:

- ToString(): Returns a formatted string like "Name: firstName lastName\nAddress: street, city\nTelephone: telephone".

Intermediate Base Class: Student (inherits from Person)

The Student class extends Person for all student types, adding academic performance attributes and an abstract grading method.

Attributes:

- test1 (double): Score for the first test.
- test2 (double): Score for the second test.

Constructors:

1. Default constructor: Calls the base Person default constructor and initializes test1 and test2 to 0.0.
2. Parameterized constructor: Calls the base Person parameterized constructor, then accepts and assigns test1 and test2.

Methods:

- abstract double ComputeGrade(): Must be overridden in derived classes to implement specific grading formulas.
- override ToString(): Extends the base ToString by appending "\nTest 1: test1, Test 2: test2\nGrade: ComputeGrade()".

Derived Student Classes

PartTimeStudent (inherits from Student)

Represents part-time undergraduate students using social security number for identification.

Attributes:

- ssNum (string): Social security number.

Constructors:

1. Default constructor: Calls the base Student default constructor and initializes ssNum to "".
2. Parameterized constructor: Calls the base Student parameterized constructor, then accepts and assigns ssNum.

Methods:

- override double ComputeGrade(): Returns 0.4 * test1 + 0.6 * test2.
- override ToString(): Returns "Part-Time Student\nSSN: ssNum\n" followed by the base student details.

UnderGradStudent (inherits from Student)

Represents full-time undergraduate students.

Attributes:

- id (string): Student ID.

Constructors:

1. Default constructor: Calls the base Student default constructor and initializes id to "".
2. Parameterized constructor: Calls the base Student parameterized constructor, then accepts and assigns id.

Methods:

- override double ComputeGrade(): Returns 0.45 * test1 + 0.55 * test2.
- override ToString(): Returns "Undergraduate Student\nID: id\n" followed by the base student details.

GradStudent (inherits from Student)

Represents graduate students with a thesis requirement.

Attributes:

- id (string): Student ID.
- thesis (string): Thesis title or description.

Constructors:

1. Default constructor: Calls the base Student default constructor and initializes id and thesis to "".
2. Parameterized constructor: Calls the base Student parameterized constructor, then accepts and assigns id and thesis.

Methods:

- override double ComputeGrade(): Returns 0.45 * test1 + 0.55 * test2 (same as UnderGradStudent).
- override ToString(): Returns "Graduate Student\nID: id\nThesis: thesis\n" followed by the base student details.

PhDStudent (inherits from Student)

Represents PhD candidates with advisor and dissertation details.

Attributes:

- id (string): Student ID.
- phDAdvisor (string): Name of the PhD advisor.
- dissertation (string): Dissertation title or description.

Constructors:

1. Default constructor: Calls the base Student default constructor and initializes id, phDAdvisor and dissertation to "".
2. Parameterized constructor: Calls the base Student parameterized constructor, then accepts and assigns id, phDAdvisor and dissertation.

Methods:

- override double ComputeGrade(): Returns 0.3 * test1 + 0.7 * test2.
- override ToString(): Returns "PhD Student\nID: id\nAdvisor: phDAdvisor\nDissertation: dissertation\n" followed by the base student details.

Intermediate Base Class: Employee (inherits from Person)

The Employee class extends Person for salaried staff, adding employment-specific attributes.

Attributes:

- department (string): Department affiliation.

Constructors:

1. Default constructor: Calls the base Person default constructor and initializes department to "".
2. Parameterized constructor: Calls the base Person parameterized constructor, then accepts and assigns department.

Methods:

- override ToString(): Extends the base ToString by appending "\nDepartment: department".

Derived Employee Classes

LabManager (inherits from Employee)

Manages university laboratories.

Attributes:

- empID (string): Employee ID.

Constructors:

1. Default constructor: Calls the base Employee default constructor and initializes empID to "".
2. Parameterized constructor: Calls the base Employee parameterized constructor and accepts empID.

Methods:

- override ToString(): Returns "Lab Manager\nEmployee ID: empID\n" followed by the base employee details.

Faculty (inherits from Employee)

Represents full-time faculty members with academic rank.

Attributes:

- empID (string): Employee ID.

- rank (string): Faculty rank (e.g., "Assistant Professor", "Associate Professor", "Professor").

Constructors:

1. Default constructor: Calls the base Employee default constructor and initializes empID and rank to "".
2. Parameterized constructor: Calls the base Employee parameterized constructor, then accepts and assigns empID and rank.

Methods:

- override ToString(): Returns "Faculty\nEmployee ID: empID\nRank: rank\n" followed by the base employee details.

Secretary (inherits from Employee)

Handles administrative secretarial duties.

Attributes:

- empID (string): Employee ID.

Constructors:

1. Default constructor: Calls the base Employee default constructor and initializes empID to "".
2. Parameterized constructor: Calls the base Employee parameterized constructor and accepts empID.

Methods:

- override ToString(): Returns "Secretary\nEmployee ID: empID\n" followed by the base employee details.

PartTimeFaculty (inherits from Employee)

Represents part-time faculty, using social security number instead of employee ID for identification.

Attributes:

- ssNum (string): Social security number.

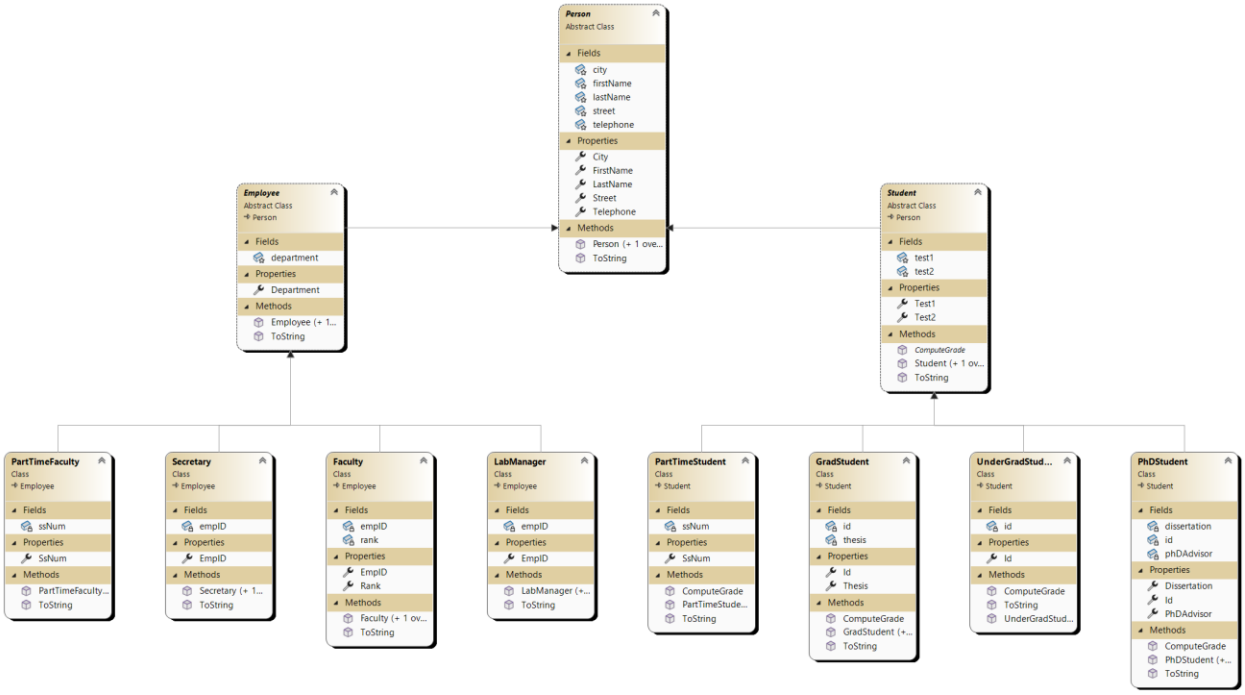
Constructors:

1. Default constructor: Calls the base Employee default constructor and initializes ssNum to "".
2. Parameterized constructor: Calls the base Employee parameterized constructor, then accepts and assigns ssNum.

Methods:

- override ToString(): Returns "Part-Time Faculty\nSSN: ssNum\n" followed by the base employee details.

UML Diagram



The UML class diagram illustrates the complete hierarchical structure of the Clark University Application system. The diagram displays the following key elements:

Inheritance Relationships:

- Person (abstract base class) serves as the root, with inheritance arrows pointing to Student and Employee intermediate classes
- Student branches into four derived classes: PartTimeStudent, UnderGradStudent, GradStudent, and PhDStudent
- Employee branches into four derived classes: LabManager, Faculty, PartTimeFaculty, and Secretary

Class Components: Each class box displays:

- Class name with "Abstract Class" notation where applicable
- Fields section listing all attributes with their data types
- Properties section showing public accessors
- Methods section including constructors and overridden methods (ToString, ComputeGrade for students)

Key Features Shown:

- Abstract classes (Person, Student, Employee) are clearly marked
- Inheritance relationships use arrows pointing from derived classes to base classes
- The ComputeGrade method appears in the Student class and is overridden in each student type
- Employee-derived classes show empID (except PartTimeFaculty which uses ssNum)
- All classes inherit common Person attributes (firstName, lastName, street, city, telephone)

The diagram effectively demonstrates the three-tier hierarchy: base class -> intermediate base classes -> concrete derived classes, showcasing the object-oriented design principles of inheritance, encapsulation, and polymorphism used throughout the system.