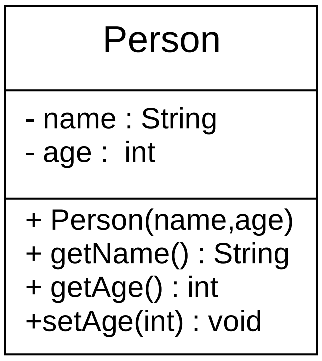
Some exercises will use class Person described below:



Exercise 1:

<u>Abstract Factory</u>

- Write a program that supports writing and reading from files and DB (Access DB using JDBC.ODBC)
- Writing to a file includes these features:
  - Defining the file name to write or read from
  - Wrapping with a buffer
  - Writing/Reading Persons
- Writing to the DB is also done in three steps:
  - Loading driver and creating connection
  - Person to DB serializer which breaks Objects into Record and vise versa
  - Writing/Reading Persons
- Client chooses to work with files or DB but once the choice was made – client code is identical in both cases. This means that and beside specifying the source (File/DB) working with the actual resource should be transparent and includes the following operations:
  - void writePerson (Person)
  - Person readPerson()
  - Person readPerson (String name)

Exercise 2:

Adapter

- Write a program that will expose the work with java.util.List collections through pop() and push() methods
- Things to do:
  - Write a class that wraps any given List collection
  - Implement two methods:
    - Push – that appends a given object
    - Pop – pulls the last object from the collection

Exercise 3:

Observer

- Write a program that scans a given text file.
- The program will notify any registered listeners with each word scanned
- The types of listeners required are:
  - Word counter – that simply counts the total words sent to it
  - Number counter – that count the total numbers of string that represents numbers (for example "345", "0")
  - Longest word keeper – which keeps the last longest word sent to it
  - Reverse word – which reverse chars order in every given word

Exercise 4:

Façade

- Add to class Person a data member named iq [IQ]
- Add to class Person another constructor that takes name,age and iq
- Add iq getters and setters
- Write a program that instantiates several persons and stores them in a file.
- Define a class that allows to do the following:
    - Check which of two persons is smarter
    - Move some IQ from one person to another and store the changes
    - Increment or reduce a person's IQ and store the changes

Exercise 5:

Composite - Lab 1

- Write a program that reflects a hierarchical file system
- Use *FSEntity* interface that defines two main file operations:
    - File/directory name
    - File/directory size
- Create *File* class to represent files
- Create *Directory* class to represent directory (empty or with files & directories in it)
    - Add file management operations to this class (add, remove, list files)
- Implement the two operations in each

Composite – Lab 2(Optional)

- Write a program that loads an XML structure into memory.
- The XML components supported by this program are:
    - Element
        - may hold inner attributes (attributes collection)
        - may be a leaf in the hierarchy
        - may hold sub elements (hold collection of elements)
        - has a print method that prints the element's name (and if there are sub-elements – also calls their print method recursively)
    - Attribute
        - Holds name and value
- Write a program that loads any given XML into an Elements tree
- Use DOM, SAX or StringTokenizer for scanning the XML input

Exercise 6:

Proxy

- Write a class that receives all the readPerson(String name) calls
- The class should delegate the request to the DB or File if no person with the matching name was already read. Otherwise it should return a cached instance of that person.

Exercise 7:

Singleton

- Create class Superman
- Since there is only one Superman in the world history – make it a singleton

Exercise 8:

Iterator

- Create an java.util.Iterator implementation that works in LIFO fashion.
- Define a class that inherits ArrayList and returns the LIFO iterator to its clients
- LIFO – Last in First Out where in – means set/add operation and out -means remove operation.

Exercise 9:

Decorator

- Create a PersonOutputStream that implements the writePerson(Person) method and can decorate any given OutputStream.
- Create a PersonOutputStream that implements the readPerson() method that returns a Person and can decorate any given InputStream.
- The PersonOutputStream decorator must check if the name of the person starts with a capital letter and if it doesn't – it should update it before writing it to the destination.
- Write a program that uses the two decorators to write and read persons to and from a file

Exercise 10:

<u>Visitor</u>

- Create an Employee class with the following attributes:
    - Name
    - Salary
    - Department
- Create a Company class that holds a collection of Employees
- Company must provide the following:
    - Total salaries computation
    - Number of employees
    - Average salary
    - Number of employees per department
    - Salary raise (by percent)
- Implement all Company activities via Visitors