CS 221
Programming Project

Clocks
(50 points)

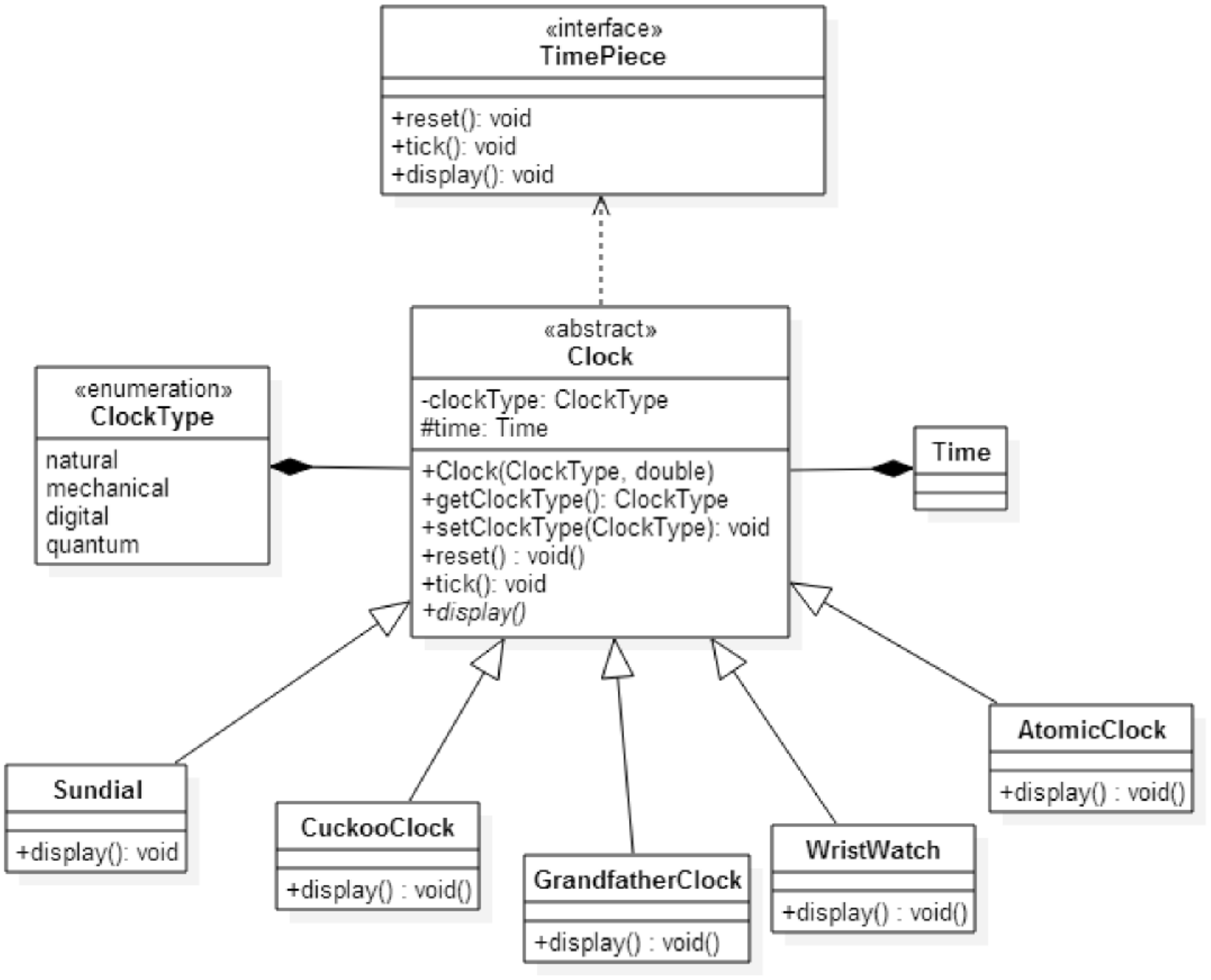
Objectives

- Implement class inheritance in Java
- Use polymorphism to store classes from the same inheritance tree in the same container
- Use of generic programming to dynamically assign stored objects

Tasks

For this project, you will write a program that simulates the running of a collection of clocks over various periods of time. Because each clock has a different amount of drift - the amount time that the clock loses as it runs - most of the clocks will display a different time at the end of a given time period.

The various types of clocks and other associated classes that you will implement are shown in the class hierarchy below:



In order to complete the project, you should:

1. **Implement the `TimePiece` interface:**

   - **Declare and document three public methods with no return values:**
     - **`reset` - resets the `TimePiece` to its starting time**
     - **`tick` - simulates one second of time passing**
     - **`display` - displays the current time**

2. **Create the abstract `Clock` class, which implements the `TimePiece` interface.**

   - **Declare `ClockType` as a public enumeration in your *Clock.java* file. Because enumerations can't be altered at runtime, it's not a violation of encapsulation to make them public.**

- Implement a constructor that takes two parameters:
    - an instance of the `ClockType` enumeration - the category of clocks that this `Clock` object falls under
    - a double representing the amount of drift for that clock - time lost per second of real time

  In the constructor, set the `Clock` object's `ClockType` and configure its `Time` variable with the given drift and a default start time of midnight.

- Implement an accessor method for private variable `clockType`. Because instance variable `time` is protected, child classes won't need an accessor method to access it.

- Implement the `reset` and `tick` methods, which are required by the `TimePiece` interface and should work the same for all descendents of `Clock`. This will require making appropriate calls to the `time` instance variable.

- Do NOT implement a `display` method, which is also required by the `TimePiece` interface. Instead, flag it with the `abstract` modifier to force children of `Clock` to implement `display` or remain abstract, themselves. Because the `Clock` class is abstract, it can have abstract methods. However, any non-abstract classes that inherit from `Clock` must implement all inherited abstract methods.

- The `Time` class has been provided and can be found in the *Time.java* file. The instance variable `time` of type `Time` should be used to represent the time stored in a given `Clock` object. Anytime a `Clock` is reset or a second passes, its `time` object should be updated appropriately. Because so much of a `Clock` object's functionality is provided by `Time`, you should carefully read through this class.

3. Implement five sub-classes of the `Clock` class:
    - `Sundial`
    - `CuckooClock`
    - `GrandfatherClock`
    - `WristWatch`
    - `AtomicClock`

   Each class should:

- **Extend the `Clock` class**
- **Create a constructor that calls the `Clock` class constructor**
- **Implement only one other method, a `display` method. The `display` method should print to standard output the time and drift of a given clock in the following format:**

  ```
          clock_type description time [YX:XX:XX],
  total drift = Z.ZZ seconds
  ```

  where `clock_type` specifies the `ClockType` the clock, `description` indicates which clock class is displaying the time, `Y` is blank if the hour doesn't have two digits, and `Z.ZZ` represents a double-precision number that may have one or more leading digits and only two decimal places.

  For example, given a `GrandfatherClock` object that has been run for one day with a drift of 30 seconds per day, the display method should print the following:

  ```
          mechanical grandfather clock time
  [11:59:30], total drift = 30.00 seconds
  ```

  Note: the actual time displayed may be off by up to a second due to rounding errors.

Below is a table summarizing information about each type of clock, including their descriptions and drifts per second:

Class	Description	Type	Drift (per second)
Sundial	sun dial	natural	0.00
CuckooClock	cuckoo clock	mechanical	0.000694444
GrandfatherClock	grandfather clock	mechanical	0.000347222
WristWatch	wrist watch	digital	0.000034722
AtomicClock	atomic clock	quantum	0.0

4. **Create a class for running a clock simulation called `ClockSimulation`.**

   - **Create a container to hold `Clock` objects using the `Bag` class in the *Bag.java* file.**

- Create one object from each of the `Clock` sub-classes with a start time of midnight and add them to your `Bag` container. These objects shouldn't be declared as variables. They should be created and stored directly in the container. This is an example of polymorphism - storing objects of different classes in a container configured to store a common ancestor class.

5. Output from the simulation should display the time of each of the clocks after each of the following time periods:

   - before the simulation begins (the starting time)
   - after one day (86,400 seconds)
   - after one week (604,800 seconds)
   - after one month (2,592,000 seconds)
   - after one year (31,536,000 seconds)

   Pay attention to output formatting - include a clear separation between each clock object and label the times (i.e. "starting time," "after 1 week," etc.) being reported.

   Be sure to reset the clocks to their start times and reset the total drift back to zero between each simulated time period.

   Because these times are in seconds and are very large numbers, you should use the `long` data type to store how much time has passed.

Files

To complete this project, you will need:

- the `Bag` class
- the `Time` class.

You will also need these exception classes, required by the `Bag` class:

- EmptyCollectionException
- ElementNotFoundException

For the other classes and the interface, you should create your own appropriately named Java files.

Grading

Points will be awarded according to the following breakdown:

Tasks	Points
Includes README file, comments	10
Table of simulation events correctly formatted	10
Correctly uses polymorphism	20
Displays correct values for each clock over each time period	10

Required Files

All the files should be well documented, to include program and function headers that use common Javadoc notation, as well as appropriate in-line comments.

Submit only the following files:

- **TimePiece.java**
- **Clock.java**
- **Sundial.java**
- **CuckooClock.java**
- **GrandfatherClock.java**
- **AtomicClock.java**
- **WristWatch.java**
- **ClockSimulation.java**
- **README**

Submission

Use the submission command given on your section's class web page from the directory containing your files.

Sample Output

Example 1:

```
$ java ClockSimulation

*** Clock Simulations, Drift Per Second: 0.0 ***

       starting  natural sun dial time [ 0:00:00], total drift = 0.00 seconds
    after 1 day  natural sun dial time [ 0:00:00], total drift = 0.00 seconds
   after 1 week  natural sun dial time [ 0:00:00], total drift = 0.00 seconds
  after 1 month  natural sun dial time [ 0:00:00], total drift = 0.00 seconds
   after 1 year  natural sun dial time [ 0:00:00], total drift = 0.00 seconds

*** Clock Simulations, Drift Per Second: 6.94444E-4 ***

       starting  mechanical cuckoo clock time [ 0:00:00], total drift = 0.00 seconds
    after 1 day  mechanical cuckoo clock time [23:59:00], total drift = 60.00 seconds
   after 1 week  mechanical cuckoo clock time [23:53:00], total drift = 420.00 seconds
  after 1 month  mechanical cuckoo clock time [23:30:00], total drift = 1800.00 seconds
   after 1 year  mechanical cuckoo clock time [17:55:00], total drift = 21899.99 seconds

*** Clock Simulations, Drift Per Second: 3.47222E-4 ***

       starting  mechanical grandfather clock time [ 0:00:00], total drift = 0.00 seconds
    after 1 day  mechanical grandfather clock time [23:59:30], total drift = 30.00 seconds
   after 1 week  mechanical grandfather clock time [23:56:30], total drift = 210.00 seconds
  after 1 month  mechanical grandfather clock time [23:45:00], total drift = 900.00 seconds
   after 1 year  mechanical grandfather clock time [20:57:30], total drift = 10949.99 seconds

*** Clock Simulations, Drift Per Second: 3.4722E-5 ***

       starting  digital wrist watch time [ 0:00:00], total drift = 0.00 seconds
    after 1 day  digital wrist watch time [23:59:57], total drift = 3.00 seconds
   after 1 week  digital wrist watch time [23:59:39], total drift = 21.00 seconds
  after 1 month  digital wrist watch time [23:58:30], total drift = 90.00 seconds
   after 1 year  digital wrist watch time [23:41:45], total drift = 1094.99 seconds

*** Clock Simulations, Drift Per Second: 0.0 ***

       starting  quantum atomic clock time [ 0:00:00], total drift = 0.00 seconds
    after 1 day  quantum atomic clock time [ 0:00:00], total drift = 0.00 seconds
   after 1 week  quantum atomic clock time [ 0:00:00], total drift = 0.00 seconds
  after 1 month  quantum atomic clock time [ 0:00:00], total drift = 0.00 seconds
   after 1 year  quantum atomic clock time [ 0:00:00], total drift = 0.00 seconds
```

Example 2:

```
$ java ClockSimulation

Times before start:
natural sun dial                time [ 0:00:00],       total drift = 0.00
mechanical cuckoo clock         time [ 0:00:00],       total drift = 0.00
mechanical grandfather clock    time [ 0:00:00],       total drift = 0.00
quantum atomic clock            time [ 0:00:00],       total drift = 0.00
digital wrist watch             time [ 0:00:00],       total drift = 0.00

After one day:
natural sun dial                time [24:00:00],       total drift = 0.00
mechanical cuckoo clock         time [23:59:01],       total drift = 60.00
mechanical grandfather clock    time [23:59:31],        total drift = 30.00
quantum atomic clock            time [24:00:00],        total drift = 0.00
digital wrist watch             time [23:59:58],        total drift = 3.00

After one week:
natural sun dial                time [24:00:00],        total drift = 0.00
mechanical cuckoo clock         time [23:53:01],        total drift = 420.00
mechanical grandfather clock    time [23:56:31],        total drift = 210.00
quantum atomic clock            time [24:00:00],        total drift = 0.00
digital wrist watch             time [23:59:40],        total drift = 21.00

After one month:
natural sun dial                time [24:00:00],        total drift = 0.00
mechanical cuckoo clock         time [23:30:01],        total drift = 1800.00
mechanical grandfather clock    time [23:45:01],        total drift = 900.00
quantum atomic clock            time [24:00:00],        total drift = 0.00
digital wrist watch             time [23:58:31],        total drift = 90.00

After one year:
natural sun dial                time [24:00:00],        total drift = 0.00
mechanical cuckoo clock         time [17:55:01],        total drift = 21899.99
mechanical grandfather clock    time [20:57:31],        total drift = 10949.99
quantum atomic clock            time [24:00:00],        total drift = 0.00
digital wrist watch             time [23:41:46],        total drift = 1094.99
```