Design Patterns Exercises

Exercise – Multiple choice

Choose the design pattern that is most suitable for the following cases.

1) We want to construct a complex object part by part
   a) Builder
   b) Factory
   c) Abstract Factory
   d) Prototype

2) There are rules about how to manage a set of objects. These can relate to the number, the lifespan and more.
   a) Singleton
   b) Object Pool
   c) Decorator
   d) Chain of Responsibility

3) When a new object is needed, clone an existing object.
   a) Mediator
   b) Interpreter
   c) Memento
   d) Prototype

4) Encapsulates that another class having the wrong interface.
   a) Proxy
   b) Facade
   c) Adapter
   d) Template

5) Don't settle for the complex interface – simplify it.
   a) Proxy
   b) Facade
   c) Adapter
   d) Template

6) Multiple wrappers for existing functionality.
   a) Chain of Responsibility
   b) Decorator
   c) Bridge
   d) Visitor

7) Choose from multiple algorithms at runtime.

a) Strategy
b) Template
c) Chain of Responsibility
d) Mediator

8) We have a consistent set of steps to follow but individual steps may have different implementations.
a) Strategy
b) Template
c) Chain of Responsibility
d) Mediator

9) Encapsulates both which services are available and the preference of which service to do the task
a) Strategy
b) Template
c) Chain of Responsibility
d) Mediator

10) Define a central class that acts as a message routing service to all other classes.
a) Strategy
b) Template
c) Chain of Responsibility
d) Mediator

11) Encapsulate the nature of a collection
a) Memento
b) Iterator
c) Observer
d) Visitor

12) Serialization in Java
a) Memento
b) Iterator
c) Observer
d) Visitor

13) Generally used between GUI class and Business logic class.
a) Memento
b) Iterator
c) Observer
d) Visitor

14) It should not be used to store global variables, but it should only be used to limit the number of objects for a class.
a) Bridge
b) Abstract Factory
c) Factory

d) Singleton

15) Java I/O uses this pattern heavily
    a) Prototype
    b) Abstract Factory
    c) Factory
    d) Decorator

16) To localize the creation of a single family of objects, we use
    a) Abstract Factory
    b) Factory
    c) Builder
    d) Prototype

17) AWT in Java for the creation of GUI uses
    a) Abstract Factory
    b) Factory
    c) Builder
    d) Prototype

18) While using JUnit testing, we normally replace heavy-duty time-consuming objects like database, network, etc with Mocks.
    a) Visitor
    b) Command
    c) Proxy
    d) Bridge

19) For undo/redo in an editor
    a) Proxy
    b) Visitor
    c) Bridge
    d) Command

20) To store Files and Directories, where a directory can contain other directories and files.
    a) Singleton
    b) Composite
    c) Chain of Responsibility
    d) Prototype

Exercise – Procedural to OO

Let's say we have an entity in the system that is responsible for monitoring the health of a connection to another machine. On a regular basis, this monitor is polled by another entity in the system, and it then checks

the connection to see if it is still up and running properly. Whether it is or not, it then sends a message elsewhere that indicates this status.

However, let's say where this status should be sent is a varying issue. Sometimes (during business hours, let's say), we want to send this information to a local display object; sometimes (after hours), we want to send this information to a pager, but in that case only if there is a failure. We do not want to wake the systems operator in the middle of the night to say that everything's fine!

The procedural approach is to create a slightly complex conditional as shown below:

```
get the status of the connection
if (it is between 8AM and 5PM)
    send the status to the local display
if (the connection failed or time between 5PM and 8AM)
        send the status to the pager
do whatever cleanup is needed and return
```

Which patterns will help you to improve this code. Assume that the condition checking needs a few lines of code and sending the status also is a significant task.

Exercise – Storage Explorer

The idea of this application came when I desperately needed an application similar to Windows Explorer that can show me the file size composition inside my storage. I want to know, for example, under the Program Files folder, which folder uses what size of space, including the numbers in percentage. I want also to know how big my mp3 total size in drive D compare to my jpg files. The information I want to see should look like this:

```
C:\Program Files:
      Microsoft          445,123 KB        43%
      Adobe              234,744 KB        25%
      Symantec            98,906 KB        10%
      ...
```

And this

```
D:\
      *.mp3              602,456 KB        47%
      *.jpg              305,830 KB        30%
      *.doc              245,355 KB        20%
      ....
```

I need the information is presented in a chart to help me visualize the numbers as well.

The application features are:

- A TreeView containing nodes that represents file system folders. The folders can be selected to see the information about file size structure within that path.
- It shows information about files size grouped by folders
- It shows information about files size grouped by file type
- The information is presented in listview, pie chart and bar chart on the right panel.

Draw the software class diagram and mention the Design patterns used.

Exercise: HR Application

This system consists of a database of the company's employees, and their associated data, such as time cards. The system must pay all employees the correct amount, on time, by the method that they specify. Also, various deductions must be taken from their pay.

- Some employees work by the hour. They are paid an hourly rate that is one of the fields in their employee record. They submit daily time cards that record the date and the number of hours worked. If they work more than 8 hours per day, they are paid 1.5 times their normal rate for those extra hours. They are paid every Friday.
- Some employees are paid a flat salary. They are paid on the last working day of the month. Their monthly salary is one of the fields in their employee record.
- Some of the salaried employees are also paid a commission based on their sales. They submit sales receipts that record the date and the amount of the sale. Their commission rate is a field in their employee record. They are paid every other Friday.
- Employees can select their method of payment. They may have their paychecks mailed to the postal address of their choice, have their paychecks held by the paymaster for pickup, or request that their paychecks be directly deposited into the bank account of their choice.
- Some employees belong to the union. Their employee record has a field for the weekly dues rate. Their dues must be deducted from their pay. Also, the union may access service charges against individual union members from time to time. These service charges are submitted by the union on a weekly basis and must be deducted from the appropriate employee's next pay amount.
- The payroll application will run once each working day and pay the appropriate employees on that day. The system will be told what date the employees are to be paid to, so it will generate payments for records from the last time the employee was paid up to the specified date.

Draw the software class diagram needed for the above application in the business layer. Mention the design patterns used.

Exercise: Game of Life

The Game of Life is not a game in the conventional sense. There are no players, and no winning or losing. Once the "pieces" are placed in the starting position, the rules determine everything that happens later. Nevertheless, Life is full of surprises! In most cases, it is impossible to look at a starting position (or *pattern*) and see what will happen in the future. The only way to find out is to follow the rules of the game.

Rules of the game: Life is played on a grid of square cells--like a chess board but extending infinitely in every direction. A cell can be ***live*** or ***dead***. A live cell is shown by putting a marker on its square. A dead cell is shown by leaving the square empty. Each cell in the grid has a neighborhood consisting of the eight cells in every direction including diagonals.

To apply one step of the rules, we count the number of live neighbors for each cell. What happens next depends on this number.

- A dead cell with exactly three live neighbors becomes a live cell (birth).
- A live cell with two or three live neighbors stays alive (survival).
- In all other cases, a cell dies or remains dead (overcrowding or loneliness).

Note: The number of live neighbors is always based on the cells ***before*** the rule was applied. In other words, we must first find all of the cells that change before changing any of them. Sounds like a job for a computer!

Play this game on the computer. What classes will be needed, to write this game? Which design patterns will be useful?

Exercise: Project – Design, Code and Unit-Test Zip File Extractor

Here we will:

- Design the first version of a ZIP file manager and extractor
- Produce a set of UML class and sequence diagrams to document your design
- Write a covering set of unit tests
- Experience working with Eclipse and JUnit

Requirements

Create an executable named `xzip` which is able to print and extract the contents of ZIP files. The program's usage from the command line should be:

```
xzip [-l | -s | -x] [-nr] <zip file name>
```

Note that the first two parameters are optional, but the file name is not. Parameters must appear in this order.

A zip file can contain files, which may be compressed (but don't have to be). In addition, a zip file can also contain directories, which may by themselves contain files and other directories. When a zip file is extracted, the directories it contains are extracted as well, and files within these directories are extracted to their correct relative location.

For the examples given below, assume that the zip file `my.zip` contains the following: A file called `mysong.mp3` which is an MP3 song, a file called `myicon.gif` which is a GIF image, and a directory called '`web`' which includes two HTML files called `index.html` and `other.html` and one directory called `backup` file which includes two other files with these same names.

The first argument of `xzip` dictates what action will be performed on the zip file:

- `-l` (long print): This action prints the names and details of the files stored in the given zip file. The long format means that in addition to the file names, more details are printed for each file. For example:

```
> java xzip -l my.zip
mysong.mp3      (7901 bytes, MP3 file)
myicon.gif      (910 bytes, 32x32 GIF image)
web             (directory, total 5 files)
   index.html       (5657 bytes, 122 lines of text)
   other.html       (6983 bytes, 208 lines of text)
   backup           (directory, total 2 files)
      index.html       (5622 bytes, 120 lines of text)
      other.html       (7501 bytes, 234 lines of text)
```

  As you can see, the details printed for each file type are different, and follow these rules:
  1. For each text file, the name of the file is printed, followed by a tab, and then the text `(N bytes, M lines of text)` where `N` is the uncompressed size of the file, and `M` is the number of lines (measured by the number of new-line characters) in the file. A text file is defined as any file whose extension is `*.txt`, `*.html` or `*.java`.
  2. For each image file, the name of the file is printed, followed by a tab, and then the text `(N bytes, WxH EXT image)` where `N` is the uncompressed size of the file, `W` is the image's width, `H` is the image's height, and `EXT` is the image file extension in uppercase. An image file is defined as any file whose extension is `*.gif` or `*.jpg`.
  3. For any other file type - any regular file not a text or zip file, such as `mysong.mp3` and `backup.zip` in the above example - print the file name, a tab, and the text `(N bytes, EXT file)` where `N` is the size of the uncompressed file and `EXT` is the file extension in uppercase.
  4. For each directory, the name of the directory is printed, followed by a tab, and then the text `(directory, total N files)` where `N` is the number of files in that directory. Each directory that the directory contains is counted as one (for the directory itself) plus the number of files inside that directory, recursively. In addition, as the example above shows, the contents of the directory are printed in the following lines, under the same rules, with an indent of three spaces relative to the indent of the directory.

- `-s` (short print): This action prints the names of the files and directories in the given zip files, but without the extra details (file size, number of text lines and so forth) which the long format provides.

Printing is equivalent to what the `-l` option outputs, including indentation and recursion into directories and zip files - only the text in parentheses for each file/directory is not printed. For example:

```
> java xzip -s my.zip
mysong.mp3
myicon.gif
web
   index.html
   other.html
   backup
      index.html
      other.html
```

- `-x` (extract): This action actually extracts the contents of the zip file into the file system. That is, for each file/directory in the zip file, a new uncompressed file/directory should be created in the file system. Files should be created in the current directory, but files that are inside directories in the zip file should be created inside these relative directories in the file system. If a file is being extracted and another file with the same name already exists, then the existing file should be overwritten.
  Upon completion, this action prints one line to the console in this format: `Extracted N files, M files failed`. `N` is the total number of files and directories that were successfully created, and `M` is the total number of files and directories that failed. In addition, one line is printed for each failed file or directory, including a detailed error message. Whenever possible, an error should not result in halting the entire program, and the program should output the error message and continue normally. For example:

  ```
  > java xzip -x my.zip
  Error: Failed to extract myicon.gif: Cannot overwrite existing file - file is in
  use
  Extracted 7 files, 1 files failed
  ```

  And the reported 7 successful files will be the following (locations are relative to the current directory):

  ```
  mysong.mp3
  web\
  web\index.html
  web\other.html
  web\backup\
  web\backup\index.html
  web\backup\other.html
  ```

The second command-line argument of `xzip` means "no recursion", and if it appears then all actions should be performed without recursion into directories. This means that only one summary line is printed for every directory in the printing actions, and that the directory is created but not populated in the extract action. For example:

```
> java xzip -l -nr my.zip
mysong.mp3      (7901 bytes, MP3 file)
myicon.gif      (910 bytes, 32x32 GIF image)
web             (directory, total 5 files)
```

```
> java xzip -x -rn my.zip
Extracted 3 files, 0 files failed
```

In this example, the 3 extracted files will be `mysong.mp3`, `myicon.gif` and `web\`.

The default action is `-l`, meaning that `xzip my.zip` is equivalent to `xzip -l my.zip`. You should print an informative usage message if the program is activated with no or illegal command-line arguments. You should print a detailed error messages and exit gracefully when a critical error occurs (the given zip file does not exist, the given zip file is corrupted and so on).

Design

While this exercise can be programmed within a single class, this won't work since this `xzip` is only a first version, so it is crucial to maintain an open mind with respect to possible future requirements. Consider the following:

1. It may be required to be able to read input format other than ZIP, such as TAR, ARJ, CAB and other archive file formats. The input does not even have to be a file: It can be the set of files of a given directory, a given FTP server address and so forth.
2. It may be required to support other file types that the long print action provides additional details about. For example, printing the image size for image files such as `myicon.gif`, or printing the song duration for music files such as `mysong.mp3`.
3. It may be required to produce the output in formats other than plain text - HTML, PDF, Word or others. It may also be required to write output in several formats at once, for example: `xzip -pdf myzipfiles.pdf -html myzipcontents.html my.zip`.
4. It may be required to modify the input zip file instead of just printing or extracting it. For example, new actions may enable adding files and directories to the zip file, changing the date and time signature of zipped files, and so on.
5. It may be required to support recursion into zip files, and not only into directories. For example, if a zip file contains another zip file, then its contents would have to be printed (recursively) like a directory's contents are printed, and when a file is extracted any zip files it contains would be extracted as well.
6. It may be required to activate all of the program's features not only from the command-line, but also from a graphical user interface, or perhaps even two user interfaces (for example, one that is a custom UI for handling zip files, and another which is fully integrated with the Windows Explorer).

You must design your program so that it is easy to add code that implements the above requirements. Assume that you are the one who will actually have to code it - that's how it usually is in "real life". For each of the above requirements write an explanation in your README file, not more than three sentences long, which explains how it should be coded.

{ If you finish early add this requirement:
It may be required to define filters on which files get printed or extracted, in addition to the -nr switch. For example, new command-line arguments can dictate that only text files should be acted on, only files that match a given pattern (such as *.cpp), and so on.

Solution: Write an Iterator for each kind of filter, whose next() method will move to the next element for which the filter is true. Such iterators are implemented as Decorators of other iterators, which easily enables to dynamically combine different filters and does not require changing or recompiling existing code.}

It is important that each solution you present will be at most three sentences long. The intention is to enforce the use of design patterns vocabulary rather than elaborating specific class and object relationships.

Code & Unit Test

This exercise intends you to divide your time equally between actual coding and between design, writing UML diagrams, and answering the above six design questions. With a proper design, this exercise is quick and simple to code. You are required to write in Java 5.0, and you may use the standard libraries to their full extent - the standard streams, strings and data structures. In particular, working with zip files is done using the `java.util.zip` package, and working with image files is done using the `javax.swing.ImageIcon` class. (While working with C++, the library at www.zlib.net may be used for compression related functions. The library at cimg.sourceforge.net may be used for image related functions.)

You are also required to use the Eclipse environment for this exercise, and are encouraged to take advantage of its editing and debugging features to their full extent.

You are required to provide class diagrams that include all your classes; there may be more than one diagram, if this is visually easier. You are also required to provide at least two sequence diagrams, depicting two interactions in your design which you consider important.

It is also required that you submit unit tests to test your work, and use the JUnit framework to do so. Organize the code such that the program source code is in one package (for example `xzip`), and the tests are in a separate package (for example `xziptest`). Each test should be self-validating - that is, know by itself whether it has passed or failed. Writing unit tests should be an integral part of coding: This is essential when code must be changed in newer versions. We recommend that you try test-first programming and in any case you will lose time if you only write all unit tests after you "finish" coding.

One metric for measuring the usefulness of a set of unit tests is called coverage, which means the percentage of your code that the unit tests actually run. Coverage of 90% or above is considered good, and you should aim to that goal.

The code you submit must be built with no compiler warnings, and pass all unit tests.

This is an advanced course, so there is emphasis is on proper design. However, as usual, you are as always expected to write clear code with a consistent style.

Exercise – Products

Draw Class diagram and mention the design patterns used for the problem below:

Your company sells both Simple Products and Product Bundles. A Simple Product is a stand-alone item, whereas a Product Bundle is a packaged collection of Simple Products. All products have a name. When you request the name of a Product Bundle, you should receive its name followed by the name of each of its constituents.
You must be able to determine the price of any product. Simple Products have a fixed unit price. The price of a Product Bundle is calculated by taking the sum of the prices of the bundle's parts. Some discount rate is applied to this total price. (for example, 10 percent for some, 15 percent for others)

Exercise – Two Problems

Describe one solution strategy that works for both problems.

A.You have a *remote* database server, and you want a *local* object to encapsulate all requests to that remote server, so that the local application programmer can treat the remote server as if it were local. How can we do this easily?

B.You have a document viewer program which may embed very large graphical images which are slow to load; but you need the viewer itself to come up very quickly when the document is selected. Therefore, images in the document should get loaded only as needed (when they come into view), not at document load time. How can we do this without complicating the design of the viewer?

Exercise – File Link

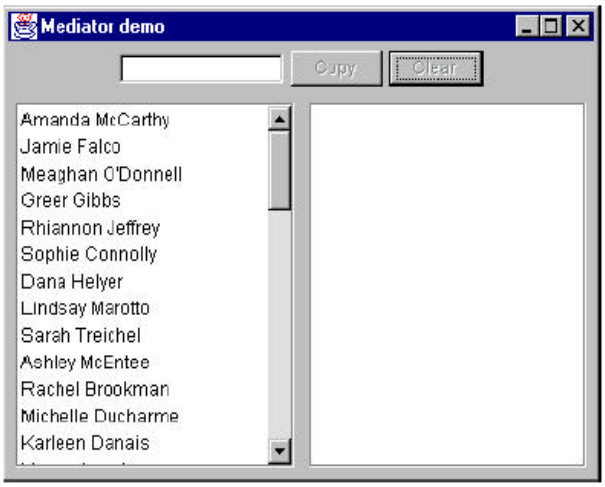
In a File Management System, there is a feature in called a Link that we wish to add to our design (like the Windows shortcut or the Macintosh alias). A Link is simply a navigational shortcut which allows a user to see a virtual copy of a file or a directory, as if it were local to the user's current location in the directory hierarchy. For most operations, the Link behaves exactly like the thing it is linked to, except that it may be deleted without deleting the actual directory or file.
Draw a class diagram for the required design

Exercise – Comany Software

Company X has a large amount of applications software written using a particular class library. Company Y that wrote the class library has now gone out of business. Company X buys a new class library from Company Z that provides the same functionality as the previous library, but unfortunately many of the classes have different interfaces. Company X does not have access to the source code for the old or the new library.
Suggest the appropriate design for the above problem.

Exercise – UI

Consider a program which has many buttons, two list boxes and a text entry field.

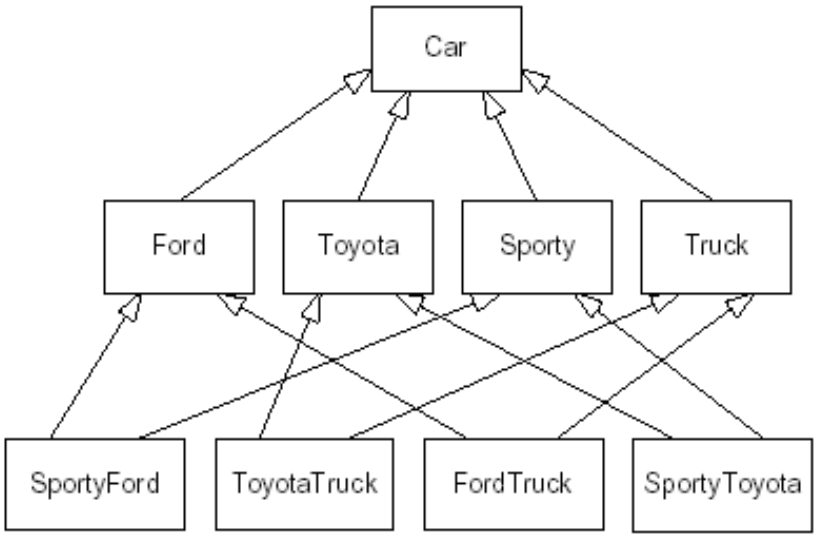


1. When you select one of the names in the left-hand list box, it is copied into the text field for editing, and the *Copy* button is enabled.
2. When you click on *Copy,* that text is added to the right hand list box, and the *Clear* button is enabled.
3. If you click on the *Clear* button, the right hand list box and the text field are cleared, the list box is deselected and the two buttons are again disabled.

User interfaces such as this one are commonly used to select lists of people or products from longer lists. Further, they are usually even more complicated than this one, involving insert, delete and undo operations as well.

Exercise – Multiple Inheritance

Simplify the following structure. You may introduce new classes.



Exercise – Workflow

Consider a workflow system supporting software developers. The system enables managers to model the process the developers should follow in terms of processes and the work products. The manager can assign specific processes to each developer and set deadlines for the delivery of each work product. The system supports several types of work products., including formatted text, picture and URLs. The manager while editing the workflow, can dynamically set the type of each work product at run time. Assuming one of design goals is to design the system so that more work product types can be added in the future, draw class diagram to represent design.

Exercise – Credit Card

You have a file that contains credit card records. Each record contains a field for the card number, the expiration date, and the name of the card holder. In your system you have the following class structure for the credit cards:

a class CreditCard,
classes VisaCC, MasterCC, AmExCC that are all subclasses of
CreditCard,

You assume more subclasses for other credit card types will be added later on.
You now have to design the structure that reads a record from the file, verifies that the credit card number is a possible account number, and creates an instance of the appropriate credit card class. What design patterns could you use for that?
Important details: Credit card numbers cannot exceed 19 digits, including a single check digit

in the rightmost position. You can also determine the card issuer based on the credit card number:

- For MasterCard, First digit is a 5, second digit is in range 1 through 5 inclusive. Only valid lenght of number is 16 digits.
- For Visa, First digit is a 4. Length is either 13 or 16 digits.
- For AmericanExpress, First digit is a 3 and second digit a 4 or 7. Length is 15 digits.
- For Discover, First four digits are 6011. Length is 16 digits.

Exercise – Book Searches

A CSV (comma separated value) file has the list of books. The first line is the header "Title, Author, Publisher, Price". The remaining lines are book; one book per line.
Now we need to do various searches on this file e.g.

1. Find the costliest 5 books.
2. Find the cheapest 5 books.
3. Find books whose title contain the substring given as input.
4. Find books whose author contain the substring given as input
5. Find books whose publisher contains the substring given as input

Design classes for above functionality. Write unit tests to prove the correctness of your code.

Exercise – Railway Ticket Booking

The cost of railway ticket depends upon the class of travel and the distance.

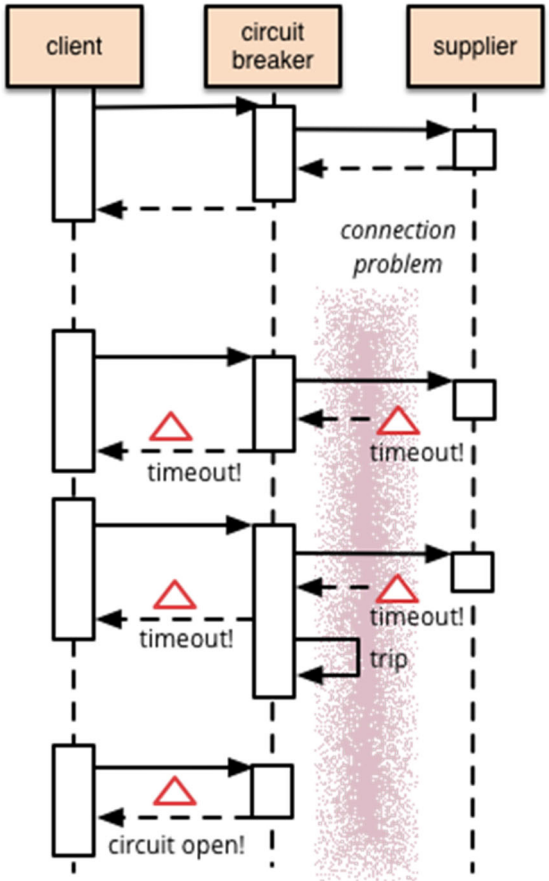
	Sleeper	AC CC	3AC	2AC	1st Class
Minimum charge	50	75	100	200	300
Kms free	20	0	0	0	0
Charge per Km in ₹	0.80	2.00	4.25	6.75	7.80

e.g. The distance from Mumbai to Pune is 180Kms. So a 2nd class Sleeper ticket will cost ₹128 i.e. (180 – 20) * 0.80. Minimum charge is not applicable, because 128 > 50.
If the travel date starts on the day of booking or next day, the ticket is treated as Tatkal. For tatkal tickets, the cost of the ticket is 3 times the normal ticket.
Senior citizen get 50% discount. No discount is given on Tatkal tickets.
The ticket prices are always rounded to nearest rupee.
Design classes for above functionality. Write unit tests to prove the correctness of your code.

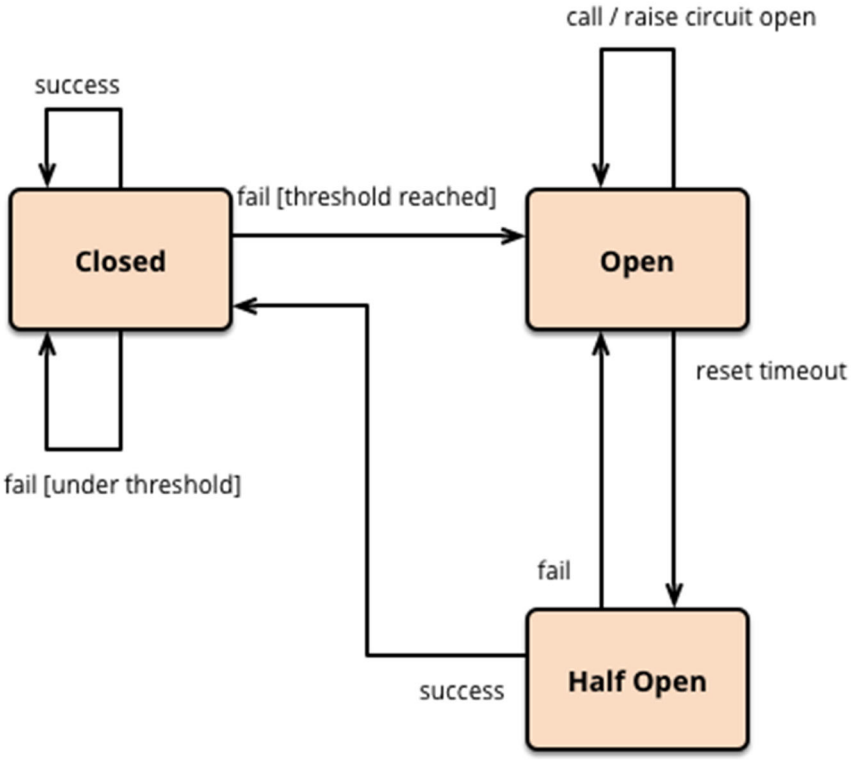
Exercise – CircuitBreaker

It's common for software systems to make remote calls to software running in different processes, probably on different machines across a network. One of the big differences between in-memory calls and remote calls is that remote calls can fail, or hang without a response until some timeout limit is reached. What's worse if you have many callers on an unresponsive supplier, then you can run out of critical resources leading to cascading failures across multiple systems. The Circuit Breaker pattern is used to prevent this kind of catastrophic cascade.

The basic idea behind the circuit breaker is very simple. You wrap a protected function call in a circuit breaker object, which monitors for failures. Once the failures reach a certain threshold, the circuit breaker trips, and all further calls to the circuit breaker return with an error, without the protected call being made at all. Usually you'll also want some kind of monitor alert if the circuit breaker trips.



This simple circuit breaker avoids making the protected call when the circuit is open, but would need an external intervention to reset it when things are well again. This is a reasonable approach with electrical circuit breakers in buildings, but for software circuit breakers we can have the breaker itself detect if the underlying calls are working again. We can implement this self-resetting behavior by trying the protected call again after a suitable interval, and resetting the breaker should it succeed.



There is now a third state present - half open - meaning the circuit is ready to make a real call as trial to see if the problem is fixed. Asked to call in the half-open state results in a trial call, which will either reset the breaker if successful or restart the timeout if not.

You have been given the tests for a working Circuit Breaker with above functionality. The skeleton code of the Circuit Breaker has been given you. You goal is to complete the code such that all tests pass. Please do not change the tests.