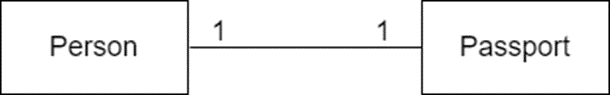
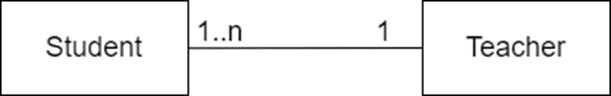
Association

* Association in Java is a connection or relation between two separate classes that are set up through their objects.
* Association relationship indicates how objects know each other and how they are using each other’s functionality.
* Association can be **one-to-one**, **one-to-many**, **many-to-one**, **many-to-many**

One to one **one-to-many Many to one**

Polymorphism

Polymorphism means ‘Different Forms of the Same Thing.

Method Overloading

* Methods MUST be in same class.
* Method Name MUST be same.
* Method Parameter MUST be different.
* Method Return Type may or may not be same.

Method Overriding

* Methods MUST be in two different classes.
* There MUST be inheritance between the two classes.
* Method Name MUST be same.
* Method Parameter MUST be same.
* Method Return Type MUST be same.

Constructor Overloading

* Constructors MUST be of same class.
* Constructor Parameter MUST be different.

Polymorphic Behavior of Objects

=

Constructor call for creation

Class name object name new key word for memory allocation

According to the Polymorphic Behavior of Objects, An Object Reference of Parent Class can hold an Object of a Child Class.

Exception

An exception is an abnormal condition that arises in a code sequence at run time. In other words, an exception is a runtime error.

* **Exception Types**
* All exception types are subclasses of the built-in class **Throwable.** Immediately below Throwable are two subclasses.
* ❖ Exception
* ❖ Error
* **Exception**
* • This class is used for exceptional conditions that user programs should catch.
* • This is also the class that you will subclass to create your own custom exception
* types.
* • There is an important subclass of Exception, called RuntimeException. Exceptions of this type are automatically defined for the programs that you write and include things such as division by zero and invalid array indexing.
* **Error**
* • Defines exceptions that are not expected to be caught under normal circumstances by your program.
* • Exceptions of type Error are used by the Java run-time system to indicate errors having to do with the run-time environment, itself.
* • Stack overflow is an example of such an error.

There are two types of exceptions:

1. Checked Exception (Compile-time Exception)

2. Unchecked Exception (Run-time Exception

Checked Exception

• A checked exception is an exception that is checked (notified) by the compiler at compilation time, these are also called as compile time exceptions

Unchecked Exception

• An unchecked exception is an exception that occurs at the time of execution. These are called as Runtime Exceptions

* **Exception-Handling Mechanism in Java**
* Java has 5 keywords for exception handling:
* ➢ **try** - Program statements that you want to monitor for exceptions are contained within a try block.
* ➢ **catch** - If an exception occurs within the try block, it is thrown. Your code can catch this exception (using catch) and handle it in some rational manner.
* ➢ **finally** - Any code that absolutely must be executed after a try block completes is put in a finally block.
* ➢ **throw** - To manually throw an exception, use the keyword throw
* ➢ **throws** - Any exception that is thrown out of a method must be specified as such by a throw’s clause

Abstraction

* **Abstract class and abstract method:**
* We can hide the details of a class or a method.
* • Abstract Class
* • Abstract Method
* • **Abstract Class:** An abstract class is a class that we cannot instantiate (cannot create object). However, we can use Object Reference of an abstract class.
* The keyword abstract is used to denote abstract class.

Interface

* **Interface:**
* Interface is just like an Abstract Class but it does not have any regular methods/non-Abstract methods in it. You cannot create any objects of an interface, but we can take object reference. Interfaces are by default public.
* **Attributes of Interface:**
* An interface may have attributes. But these attributes are by default public, static and final all at the same time. An interface does not have any constructors. All the methods of an interface are by default public and abstract.
* **Methods of Interface:**
* As all the methods are by default abstract, they do not have any body/implementations. However, if we want to give body to any method of an interface, we have to declare the method as static.
* The keyword interface is used to denote interface.

*User Defined Package*

* A **package** is a grouping of related types providing access protection and name space management.
* **Why We Need Package**
* **1. Preventing naming conflicts.** Till now, we were using **default package**, which has no name. That is why, a unique name had to be used for each class to avoid name collisions. For example, we cannot declare employee class twice in default package or **unnamed package**. But using package we can keep both employee class in separate package.
* 2. Making searching/locating and usage of classes and interfaces easier.
* 3. **Providing controlled access**. Protected and default have package level access control. A protected member is accessible by classes in the same package and its
* subclasses. A default member (without any access specifier) is accessible by classes in the same package only.

Types of Package

* There are two types of packages:
* 1. **Built-in package**: Already defined packages like java.io.\*, java.lang.\* etc. are known as built-in packages.
* 2. **User defined package**: The package we create is called user-defined package.

Creating a file

* How to create a file:
* To create a file in Java, you can use the createNewFile() method. This method returns a boolean value: true if the file was successfully created, and false if the file already exists. The method is enclosed in a try...catch block. This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason)
* **Writing in a file:**
* We use the File Writer class together with its write () method to write some text to the file we created. Note that when you are done writing to the file, you should close it with the close () method.
* **Reading from file:** We create object of File Reader class to read some text from a file we created. We create Buffered Reader object using the File Reader class object to read the file content. Note that when you are done reading to the file, you should close it with the close () method