1. **Explain inheritance**. -- Inheritance is a mechanism wherein a new class is derived from an existing class. In Java, classes may inherit or acquire the properties and methods of other classes.

A class derived from another class is called a subclass, whereas the class from which a subclass is derived is called a superclass. A subclass can have only one superclass, whereas a superclass may have one or more subclasses.

1. **Explain polymorphism with example -- -** Polymorphism is one of the [OOPs](https://beginnersbook.com/2013/04/oops-concepts/) feature that allows us to perform a single action in different ways. For example, lets say we have a class Animal that has a method sound(). Since this is a generic class so we can’t give it a implementation like: Roar, Meow, Oink etc. We had to give a generic message

public class Animal{

...

public void sound(){

System.out.println("Animal is making a sound");

}

}

Now lets say we two subclasses of Animal class: Horse and Cat that extends (see [Inheritance](https://beginnersbook.com/2013/03/inheritance-in-java/)) Animal class. We can provide the implementation to the same method like this:

public class Horse extends Animal{

...

@Override

public void sound(){

System.out.println("Neigh");

}

}

and

public class Cat extends Animal{

...

@Override

public void sound(){

System.out.println("Meow");

}

}

As you can see that although we had the common action for all subclasses sound() but there were different ways to do the same action. This is a perfect example of polymorphism (feature that allows us to perform a single action in different ways). It would not make any sense to just call the generic sound() method as each Animal has a different sound. Thus we can say that the action this method performs is based on the type of object.

*-🡪🡪 Polymorphism is the capability of a method to do different things based on the object that it is acting upon. In other words, polymorphism allows you define one interface and have multiple implementations.*

**Types of polymorphism in java- Runtime and Compile time polymorphism**

1. **Static Polymorphism** also known as compile time polymorphism  
   2) **Dynamic Polymorphism** also known as runtime polymorphism

## Compile time Polymorphism (or Static polymorphism)

Polymorphism that is resolved during compiler time is known as static polymorphism. Method overloading is an example of compile time polymorphism.

## Runtime Polymorphism (or Dynamic polymorphism)

It is also known as Dynamic Method Dispatch. Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime, that’s why it is called runtime polymorphism.

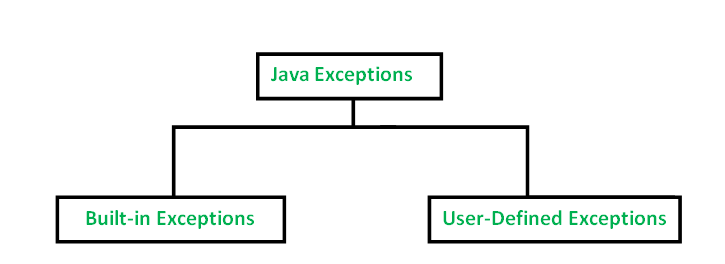
1. What is an Exception?

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

**Error vs Exception**

**Error:**An Error indicates serious problem that a reasonable application should not try to catch.  
**Exception:**Exception indicates conditions that a reasonable application might try to catch.

# Types of Exception in Java with Examples



Built-in exceptions are the exceptions which are available in Java libraries.

**Arithmetic Exception**

**ArrayIndexOutOfBoundException**

**ClassNotFoundException**

**NullPointerException**

**User-Defined Exceptions**

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called ‘user-defined Exceptions’.

Exception can handle RUNTIME EXCEPTION AS WELL.

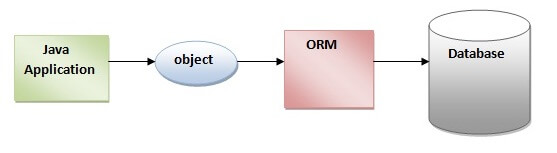
## **Hibernate Framework**

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

Hibernate framework provides some **abstraction layer** means programmer dont have to worry about the implementations, Hibernate do implementations for you internally like **Establishing a connection with the database, writing query to perform CRUD operations etc**.

1. **What is an ORM ?**

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



## **What is JPA?**

Java Persistence API (JPA) is a Java specification that provides certain functionality and standard to ORM tools. The **javax.persistence** package contains the JPA classes and interfaces.

**Need of Hibernate Framework**

Hibernate is used to overcome the of limitations of JDBC like:

1. JDBC code is dependent upon the Database software being using. i.e. Our persistence logic is dependent because of using JDBC. Here we are inserting a record into Employee table but our query is Database software dependent i.e. Here we are using MySQL. But if we change our Database then this query wont work.
2. If working with JDBC, changing of Database in middle of the project is very costly.
3. JDBC code is not portable code across the multiple database softwares.
4. In JDBC, Exception handling is mandatory. Here We can see that we are handling lots of Exception for connection.
5. While working with JDBC, There is no support Object level relationship.
6. In JDBC, there occurs a Boiler plate problem i.e. For each and every project we have to write the below code.That increases the code length and reduce the readability.