1. What is Checked Exception?

checked exceptions denote error scenarios which are outside the immediate control of the program. They occur usually interacting with outside resources/network resources e.g. database problems, network connection errors, missing files etc.

Checked Exceptions are subclass of Exception.

Example :

```
public static void main(String[] args)
{
   try
   {
      FileReader file = new FileReader("somefile.txt");
   }
   catch (FileNotFoundException e)
   {
      //Alternate logic
      e.printStackTrace();
   }
}
```

2. What is Unchecked Exception?

the occurrences of which are **not checked by the compiler**. They will come into life/occur into your program, once any buggy code is executed.

A method is not forced by compiler to declare the unchecked exceptions thrown by its implementation. Generally, such methods almost always do not declare them, as well.

Unchecked Exceptions are subclasses of **RuntimeException.**

```
public static void main(String[] args)
{
   try
   {
      FileReader file = new FileReader("pom.xml");

      file = null;

      file.read();
   }
   catch (IOException e)
   {
      //Alternate logic
```

```
    e.printStackTrace();
} }
```

3. Which is the base class of all exception and error.

The base class of all things that can be thrown is Throwable (not Exception).
Under Throwable are two subclasses: Exception and Error.

4. Can we write only try block without catch and finally blocks?

this is a feature in Java 7 and beyond. try with resources allows to skip writing the finally and closes all the resources being used in try-block itself.

```
static String readFirstLineFromFile(String path) throws IOException {
   try (BufferedReader br =
              new BufferedReader(new FileReader(path))) {
      return br.readLine();     }}
```

In this example, the resource declared in the try-with-resources statement is a BufferedReader. The declaration statement appears within parentheses immediately after the try keyword. The class BufferedReader, in Java SE 7 and later, implements the interface java.lang.AutoCloseable. Because the BufferedReader instance is declared in a try-with-resource statement, it will be closed regardless of whether the try statement completes normally or abruptly (as a result of the method BufferedReader.readLine throwing an IOException).

Prior to Java SE 7, you can use a finally block to ensure that a resource is closed regardless of whether the try statement completes normally or abruptly.

5. differentiate between error and exception?

“Error” is a critical condition that cannot be handled by the code of the program.

Exception” is the exceptional situation that can be handled by the code of the program.

The significant difference between error and exception is that an **error** is caused due to lack of system resources, and an **exception** is caused because of your code.

6. Rethrow an Exception in java?

Sometimes we may need to rethrow an exception in Java. If a catch block cannot handle the particular exception it has caught, we can rethrow the exception. The rethrow expression causes the **originally thrown object to be rethrown.**

```
import java.util.Scanner;

public class Test

{

        public static void main(String args[])

        {

          try

           {

                fun();

            }

           catch(IllegalArgumentException e)

                {

                System.out.println("value entered is not feable");

                }

                stativ void fun()

                {

                        try

                        {

                        Scanner sc = new Scanner();

                        int val= sc.nectInt();

                        if(val<0)

                        {
```

```
                throw new IllegalArgumentException();
            }
            catch(IllegalArgumentException e)
            {
             throw e;
            }
    }
    }
    }
```