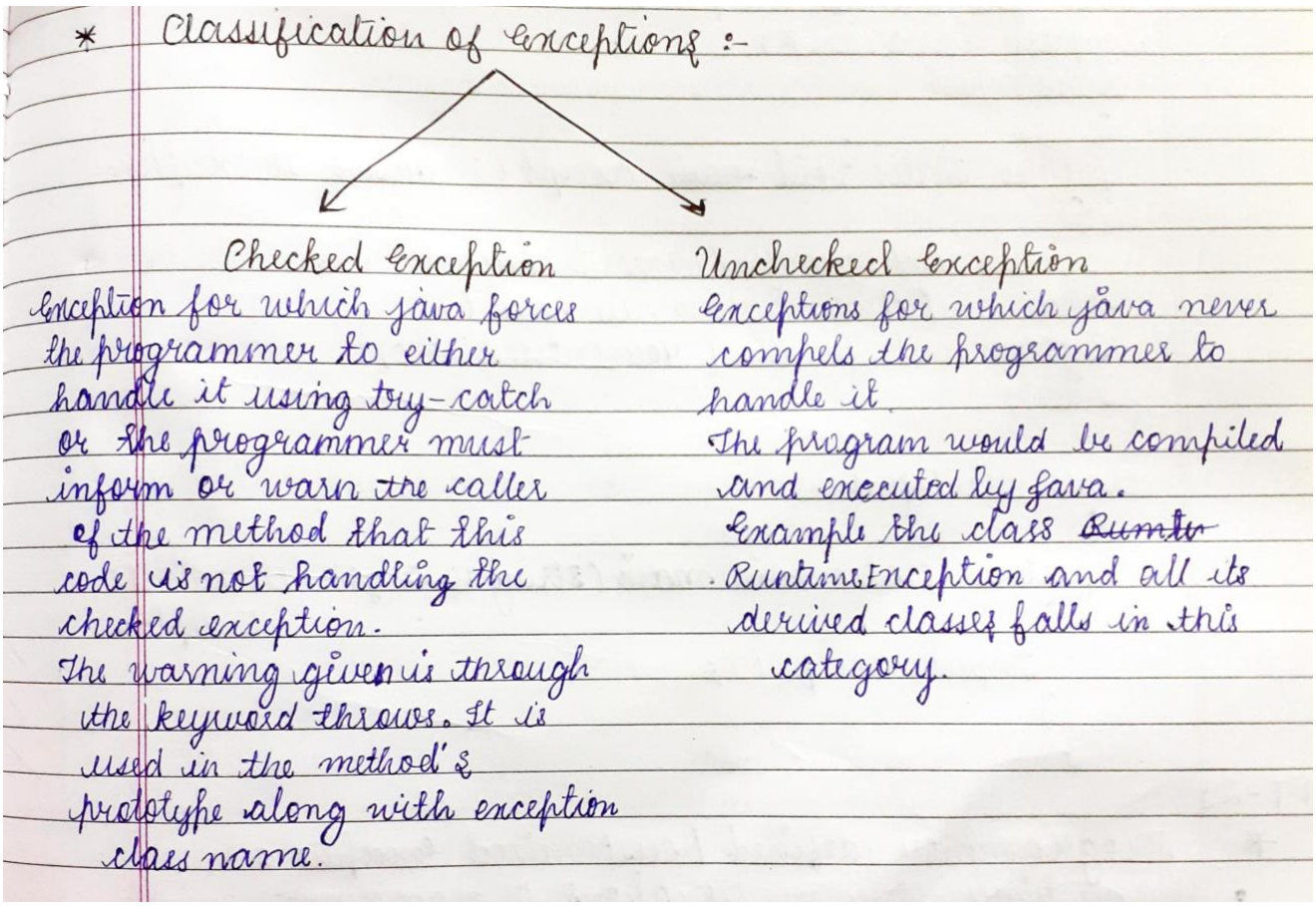
- **Classification of Exceptions:-**



- **E.g. of unchecked exception**

```
public void fun1()
{
int a ;
a = 10/0;
System.out.println(a);
}
```

Will 100% compile and java will handle unchecked exception on it's own.

- **Checked Exception:-** This in java is known as handle or declare rule.
  The caller also has 2 options, either use try-catch or the keyword throws.
  We can use throws keyword to delegate the responsibility of exception handling to the caller (It may be a method or JVM) then caller method is responsible to handle that exception.
  If every method use throws, then ultimately the responsibility goes to JVM which will follow the standard mechanism of handling the exception.
  All exceptions which are not derived classes of Run-time exception fall in this category like SQLException, IOException etc.
  E.g. public void fun2()
  {
  //code which can throw checked Exception
  }
  Will not compile

  Either handle or Declare rule

  Handle :
  public void fun2()
  {
  try
  {
  //code which can generate checked Exception

```
}
catch(.....)
{
.
}
}
```

Declare rule :

```
public void fun2() throws <checkedExceptionName>
{
//code which can throw checked exception
}
public void display() throws <checkedExceptionName>
{
fun2();
}
```

Code for checked Exception:-

```
import java.io.*;
class Input
{
public static void accept() throws IOException
{
System.out.println(“Enter a character: “);
```

```
char c = (char) System.in.read();
System.out.println("You entered: "+c);
}
}
class UseInput
{
public static void main(String [] args) throws IOException
{
Input.accept();
}
}
```

- **Programmer defined/customized exceptions :-** Many times, in some situations a programmer might not find any pre-defined exception class to be used with throw.

E.g. In case of a banking application, the method withdraw has some minimum limit like 500, else an exception will generate. In this case, there is no predefined java's exception class.

So, java advises us to design our own exception classes. Such classes are called Customized exception class.

To create an exception class we need to follow some steps, which are as follows :-

1. Inherit or extend any of the pre-defined exception class in our own exception class. Generally we inherit Exception class.
2. Provide a parametrized constructor so that exception message can be set and passed on to parent class's constructor.

E.g. import java.util.*;

```
class InvalidNumeratorException extends Exception
{
public InvalidNumeratorException (String msg)
{
super(msg);
}
}
class Test
{
public static void main(String  [ ] args)
{
Scanner kb = new Scanner(System.in);
System.out.println(“Enter two numbers: “);
try
{
```

```
int a = kb.nextInt();
int b = kb.nextInt();
if(a<=0)
{
throw new InvalidNumeratorException(“Numerator should be positive");
}
int c = a/b;
System.out.println(“Division is: “+c);
}
catch(ArithmeticException ex)
{
System.out.println(ex.getMessage());
}
catch(InvalidNumeratorException ex)
{
System.out.println(ex.getMessage());
}
catch(InputMismatchException ex)
{
System.out.println(“Please input digits only”);
}
```

```
}
}
```