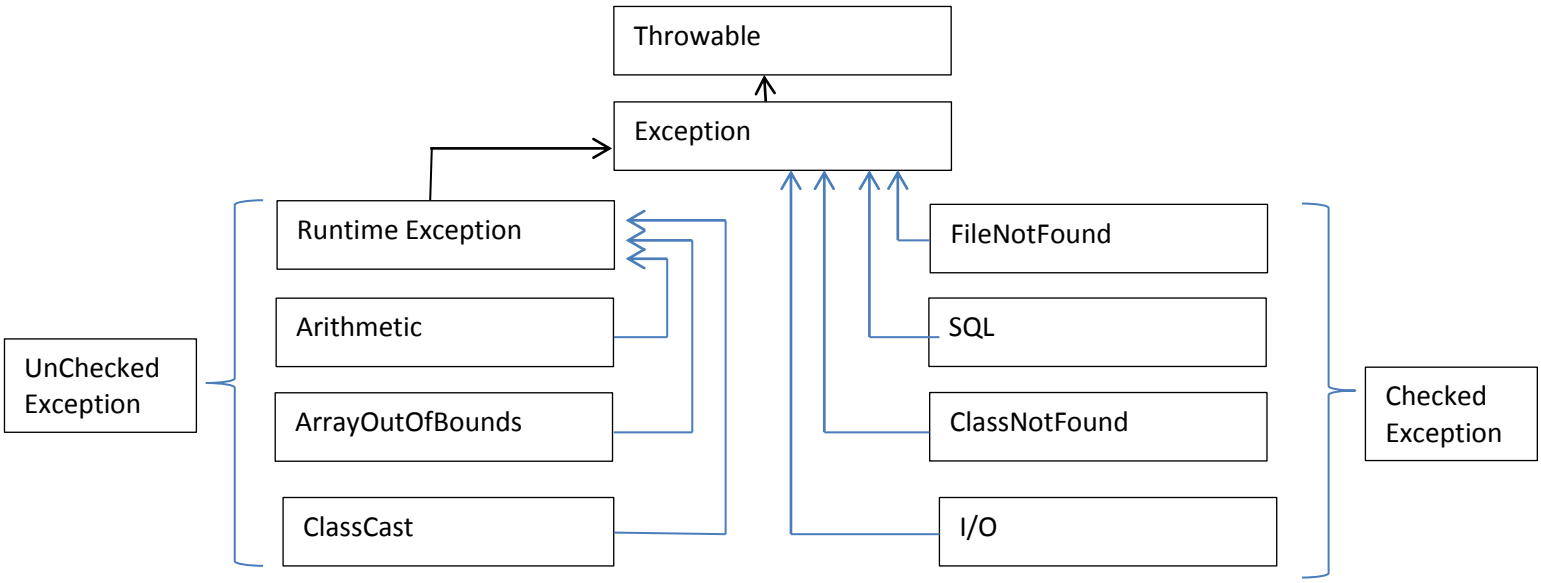
Exception Handling

Exception is an event which occurs to interrupt normal execution flow of the program. Exception can be handled with help of try and catch block.

Try block is used to declare the exception statements. Catch is used to handle the exception.

Exception Hierarchy:



Throwable is root class for all exception classes. Exception is subclass of throwable class.

There are two types of exception:

1. Checked exception
2. Unchecked exception

Checked Exception:

Compile time exception occurring at compiletime. These are also called as compile time exception. Checked exception are subclass of exception class.

Unchecked Exception:

Runtime exception occurring at runtime. These are also called as runtime exception. Unchecked exception are subclass of runtime exception and runtime exception is subclass of Exception class.

Example for try catch block:

```
public static void main(String[] args)
{
	Scanner sc=new Scanner(System.in);
	System.out.println("Main method is running");
	System.out.println("Enter the number ");
	int x=sc.nextInt();
	int y=sc.nextInt();
	try
	{
	int z=x/y;
	System.out.println(z);
	}
	catch(ArithmeticException e)
	{
		System.out.println("Number cannot be divided by zero");
	}
	System.out.println("Main method is ended ");
}
```

ArrayOutOfBoundException

```
public static void main(String[] args)
{
	System.out.println("Main is runnung");
	char ch[]=new char[3];
	ch[0]='a';

	try
	{
		ch[3]='b';
	}
	catch(ArrayIndexOutOfBoundsException e)
	{
		System.out.println("Array out of bound");
	}
	System.out.println("Main is ending");
}
```

1. After exception none of the statements in try block is executed, control once given to catch block will not return to try block.

```
public static void main(String[] args)
	{
	Scanner sc=new Scanner(System.in);
	System.out.println("Main method is running");
	System.out.println("Enter the number ");
	int x=sc.nextInt();
	int y=sc.nextInt();
	try
	{
	int z=x/y;
	System.out.println(z);
	System.out.println("Try block is executed");
	}
```

```
        catch(ArithmeticException e)
        {
                System.out.println("Number cannot be divided by zero");
        }
        System.out.println("Main method is ended ");
        }
```

Output:

```
Main method is running
Enter the number
10
0
Number cannot be divided by zero
Main method is ended
```

2. Since there is no exception, try block is executed but catch block is not executed.

```
        public static void main(String[] args)
        {
        Scanner sc=new Scanner(System.in);
        System.out.println("Main method is running");
        System.out.println("Enter the number ");
        int x=sc.nextInt();
        int y=sc.nextInt();
        try
        {
        int z=x/y;
        System.out.println(z);
        System.out.println("Try block is executed");
        }
        catch(ArithmeticException e)
        {
                System.out.println("Number cannot be divided by zero");
        }
        System.out.println("Main method is ended ");
        }
```

```
Main method is running
Enter the number
10
2
5
Try block is executed
Main method is ended
```

Finally Block

Finally block must be declared using keyword finally. Statements declared inside finally block will be executed always whether we handle exception or not in the program. Finally must follow try block or catch block.

```
        public static void main(String[] args)
        {
        Scanner sc=new Scanner(System.in);
        System.out.println("Main method is running");
        System.out.println("Enter the number ");
        int x=sc.nextInt();
```

```
int y=sc.nextInt();
try
{
int z=x/y;
System.out.println(z);
System.out.println("Try block is executed");
}
catch(ArithmeticException e)
{
        System.out.println("Number cannot be divided by zero");
}
finally
{
        System.out.println("Finally is running");
}
System.out.println("Main method is ended ");
}
```

```
Main method is running
Enter the number
10
2
5
Try block is executed
Finally is running
Main method is ended
```

When catch block is executed

```
Main method is running
Enter the number
10
0
Number cannot be divided by zero
Finally is running
Main method is ended
```

Finally Following try block:

```
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("Main method is running");
System.out.println("Enter the number ");
int x=sc.nextInt();
int y=sc.nextInt();
try
{
int z=x/y;
System.out.println(z);
System.out.println("Try block is executed");
}
finally
{
```

```
            System.out.println("Finally is running");
        }
        System.out.println("Main method is ended ");
        }
```

```
Main method is running
Enter the number
10
0
Finally is running
```

Multiple catch blocks and single try block

```
public static void main(String[] args)
        {
        try
        {
        int z=10/0;
        System.out.println(z);
        char a[]=new char[3];
        a[0]='a';
        a[3]='b';
        System.out.println("Try block is executed");
        }
        catch(ArithmeticException e)
        {
                System.out.println("Arithmetic exception block");
        }
                catch(ArrayIndexOutOfBoundsException e)
        {
                System.out.println(" ArrayOutOfBoundException block");
        }
        finally
        {
                System.out.println("Finally is running");
        }
        System.out.println("Main method is ended ");
        }
```

We can declare multiple catch block and single try block, but we cannot declare multiple try block and single catch block

Try catch illegal blocks

Try
{
 	
}
// cannot have only try block
	Catch( )
	{
	
	}
	// only catch block cannot be declared, one try block compulsory
		Finally( )
		{
		
		}
		// only finallyblock cannot be declared, one try block compulsory
			Finally( )
			{
			}
			Finally()
			{
			}
			// no multiple finally block

Finally should de declared after catch, else error is displayed.

Throwable class

Throwable is root class for all exception class. Throwable class is defined in java.lang package. Throwable class methods.

1. Public String getMessage()

getMessage is a non-static method declared in throwable class. Throwable class return reason for exception in String format.

2. Public void printStackTrace();

printStackTrace() is a non-static method declared in thrwoable class. printStackTrace will display exception details. Exception details include name of exception, reason for exception and stack information.

Example for getMessage()

```
public static void main(String args[])
{
	try
	{
		int x=10/0;
	}
	catch(ArithmeticException e)
	{
		String st=e.getMessage();
		System.out.println(st);
	}
}
```

Example for printStackTrace()

```
	public static void main(String args[])
	{
		try
		{
			int x=10/0;
			System.out.println(x);
		}
		catch(ArithmeticException e)
		{
			e.printStackTrace();
		}
	}
```

Output:

```
java.lang.ArithmeticException: / by zero
	at Exception5.ExceptionMain5.main(ExceptionMain5.java:9)
```

Try catch can be added either in separate method or within main method, where function is called.

Calling try catch inside method:

```
public static void m1()
	{
		try
		{
			int x=10/0;
```

```
		}
		catch(ArithmeticException e)
		{
			e.printStackTrace();
		}
	}

public static void main(String args[])
	{
		m1();
	}
```

Calling try catch within main calling method

```
	public static void m1()
	{
		int x=10/0;
	}

	public static void main(String args[])
	{
		try
		{
			m1();
		}
		catch(ArithmeticException e)
		{
			e.printStackTrace();
		}
	}
```

Exception class object can hold any exception either checked or unchecked object, but Runtime exception object can only hold checked exception.

Example for handling multiple exception using only single catch block.

```
public static void main(String[] args)
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the two numbers");
		try
		{
			int x=sc.nextInt();
			int y=sc.nextInt();
			int z=x/y;
			System.out.println(z);
		}
		catch(Exception e)
		{
			System.out.println("Exception handled");
		}
```

Runtime as well as Compiletime exce
ptions are handled.

Nested Try catch block

```
public static void main(String[] args)
{
	try
	{
		try
		{
			int a[]=new int[3];
			a[3]=10;
		}
		catch(Exception e)
		{
			System.out.println("A");
		}
	int z=10/0;

		}
	catch(Exception e)
	{
		System.out.println("B");
	}
}
```

UserDefined Exception:

Exception created ny user is known as user defined exception. User can create both checked and unchecked exception. User can create unchecked exception by extending Exception class and user can create checked exception by extending runtimeexception class.

```
public class UserDefinedException2 extends RuntimeException
{

}


public static void access(int age, String name)
	{
		try
		{
			if(age<18)
			{
				throw new UserDefinedException2();
			}
		}
			catch(UserDefinedException2 e)
			{
				System.out.println("Invalid age");
			}
		if(age>18)
		{
			System.out.println(name+" is elligible to cast the vote");
		}
	}
```

```
public static void main(String[] args)
{
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the value for Name and age");
        String name=sc.next();
        int age=sc.nextInt();
        access(age,name);
}
```