Throw and Throws

Throw-

- The throw keyword is used to create our own Exception object and handover it to JVM manually.
- It is used for custom exception.
- After throw statement, we can't take any other statement as it will become unreachable.

Example:

- throw new InsufficientFundException ();

Throws-

Throws keyword is used to declare the exception with method.
It is used to delegate the responsibility of exception handling to the caller method. Then caller method is responsible to handle that exception.

Note:

- Hence the main objective of "throws" keyword is to delegate the responsibility of exception handling to the caller method.
- "throws" keyword required only checked exceptions. Usage of throws for unchecked exception there is no use.
- "throws" keyword required only to convince complier. Usage of throws keyword doesn't prevent abnormal termination of the program.
- **Hence recommended to use try-catch over throws keyword.**
- We can use throws keyword only for constructors and methods but not for classes.

Example-

```
void m1()throws IOException{

   m2();

}
```

Note:

In our program with in the try block, if there is no chance of rising an exception then we can't right catch block for that exception otherwise we will get compile time error saying exception XXX is never thrown in body of corresponding try statement. But this rule is applicable only for fully checked exception.

How to create the custom exception in java

We can create our own Exception that is known as custom exception or user-defined exception.

By the help of custom exception, you can have your own exception and message.

Example-1- Scenario

Sometimes it is required to develop meaningful exceptions based on application requirements. For example suppose you have one savings account in SBI Bank and you have 50000 in your account. Suppose you attempted to withdraw 60000 from your account. In java you can handle. You need to display some error message related to insufficient fund.

Steps to create the user defined exception

1. Create the new class.
2. The user defined exception class must extend from java.lang.Exception or java.lang.RunTimeException class.
3. While creating custom exception, prefer to create an unchecked, Runtime exception than a checked exception.
4. Every user defined exception class in which parametrized Constructor must called parametrized Constructor of either java.lang.Exception or java.lang.RunTimeException class by using super(string parameter always).
5. It is highly recommended to maintain our customized exceptions as unchecked by extending RuntimeException. We can catch any Throwable type including Errors also.

```
package com.velocity;
```

```
public class InsufficientFundException extends
RuntimeException {

	private String message;

	public InsufficientFundException(String message) {
		//this.message = message;
		super(message);
	}
}


package com.velocity;

public class Account {

	private int balance = 3000;

    public int balance() {
        return balance;
    }

    public void withdraw(int amount) {
        if (amount > balance) {
            throw new
InsufficientFundException("Insufficient balance in your
account..");
        }
        balance = balance - amount;
    }
}


package com.velocity;

public class MainTest {

	public static void main(String[] args) {
		Account account = new Account();
		System.out.println("Current balance : " +
account.balance());
		account.withdraw(3500);
		System.out.println("Current balance : " +
account.balance());
	}
}
```

Example-2

```
package com.test;

class Test extends Exception {

	public Test(String s) {
		super(s);
	}

}

package com.test;

public class Test1 {

	public static void main(String[] args) {

		try {

			throw new Test("Invalid input");
		} catch (Exception e) {
			e.getMessage();
		}
	}
}
```

Question. What is the difference Between Catch and Finally in java?

Sr. No.	Catch	Finally
1	Catch block handles the error when it occurs in try block	There is no need of exception thrown by try block
2	Catch block is executed only when the if exception is thrown by try block, otherwise it is not executed	Finally block is always executed whether exception occurs or not
3	We can use multiple catch block for only one try block	Only one finally block is used for one try block
4	We can handle multiple exceptions by using catch blocks	It is not for exception handling

Q. 1

```
int m1() {
   try {
    return 10;
   } catch (Exception e) {
     return 20;
   } finally {
     return 30;
   }
}
```

return-30

Even though return statement present in try or catch blocks first finally will be executed and after that only return statement will be considered. i.efinally block dominates return statement.
If return statement present try, catch and finally blocks then finally block return statement will be considered

Q 2

```
public class FinallyTest {
int m1() {
  try {
    return 10;
  } catch (Exception e) {
     return 20;
  } finally {
     return 30;
  }
  return 40;
  }
}
```

It will not compile, unreachable code return 40.

Self Assignment:

What is the difference between Checked Exception and Unchecked Exception keyword?

What is the difference between throw and throws?

What is try with resources? (version 1.7 onwards)

What is multiple catch block (version 1.7 onwards)

Note:

While overriding if the child class method throws any checked exception compulsory the parent class method should throw the same checked exception or its parent otherwise we will get compile time error.
But there are no restrictions for un-checked exceptions.

Case	**Valid / Invalid**
1) Parent : public void methodOne() throws Exception **Child : public void methodOne()**	Valid
2) Parent : public void methodOne() **Child : public void methodOne() throws Exception**	Invalid
3) Parent : public void methodOne() throws Exception **Child : public void methodOne() throws Exception**	Valid
4) Parent : public void methodOne() throws IOException **Child : public void methodOne() throws Exception**	Invalid
5) Parent : public void methodOne() throws IOException **Child : public void methodOne() throws EOFException, FileNotFoundException**	Valid
6) Parent : public void methodOne() throws IOException **Child : public void methodOne() throws EOFException, InterrueptedException**	Invalid
7) Parent : public void methodOne() throws IOException **Child : public void methodOne() throws EOFException, ArithematicException**	Valid
8) Parent : public void methodOne() **Child : public void methodOne() throws NullPointerException, ArithematicException, RuntimeException**	Valid