Using `throws`

By Rahul Barve

Using **`throws`**

- If several methods of a class are probable to fire an exception, it becomes difficult to manage writing `try-catch` constructs in each method.
- This can be simplified by using `throws`.

Using `throws`

- Used by method and constructor definitions which may fire exceptions but not willing to handle.
- Instructs the compiler to enforce the calling program to handle the exception (Checked Exceptions only).

Using **`throws`**

```
public void readFile(String fileName) throws
FileNotFoundException {
     //Statements
}
public void openFile(String fileName) {
     readFile(fileName); //ERROR
}
```

Using `throw`

Using **`throw`**

- In most cases, `JRE` is responsible for firing an exception; but sometimes it might be necessary to fire an exception forcefully.
- This can be accomplished by using `throw` clause.

Using **`throw`**

- Syntax: `throw <Throwable>`
- E.g.

```
if(<condition>){
    Exception ex = new Exception();
    throw ex;
}
```

Using `throw`

- Sometimes, it becomes necessary to create a domain specific exception and throw it explicitly.
- Such exceptions are known as User Defined exceptions.

Using `throw`

- User defined exceptions are generally customized by creating a class that inherits either `Exception` or `RuntimeException`.
- E,g,

```
 public    class    LowBalanceException
extends Exception {…}
```

Using `throw`

- Once, a user defined exception class is created it can be used to raise an exception forcefully depending upon the condition.

Using throw

```
public  void  withdraw(float  amount)
throws LowBalanceException {
    if(balance < amount){
        String msg = “Low Balance!!”;
        LowBalanceException lx =
        new LowBalanceException(msg);
        throw lx;
    }
}
```