JA111

Day-11

Regular Expressions

A regular expression is a sequence of characters that forms a search pattern. When you search for data in a text, you can use this search pattern to describe what you are searching for. A regular expression can be a single character, or a more complicated pattern.

The main important application areas of Regular Expression are:

1. **To implement validation logic.**
2. **To develop Pattern matching applications.**
3. **To develop translators like compilers, interpreters etc.**

Regular Expression types- ^regex, regex$, X|Z, XZ, [abc], [abc][vz], [a-z], [^abc] etc.
Qualifiers: ., *, +, ?, {X} , {X,} , {X,Y}

Classes related to Pattern Matching

❑ **Pattern class**

static Pattern compile(String regex) **Matcher matcher(CharSequence input)**

❑ **Matcher class**

boolean matches() **boolean find()** **int start()**
int end() **String group()**

❑ **PatternSyntaxException**

Exception Handling

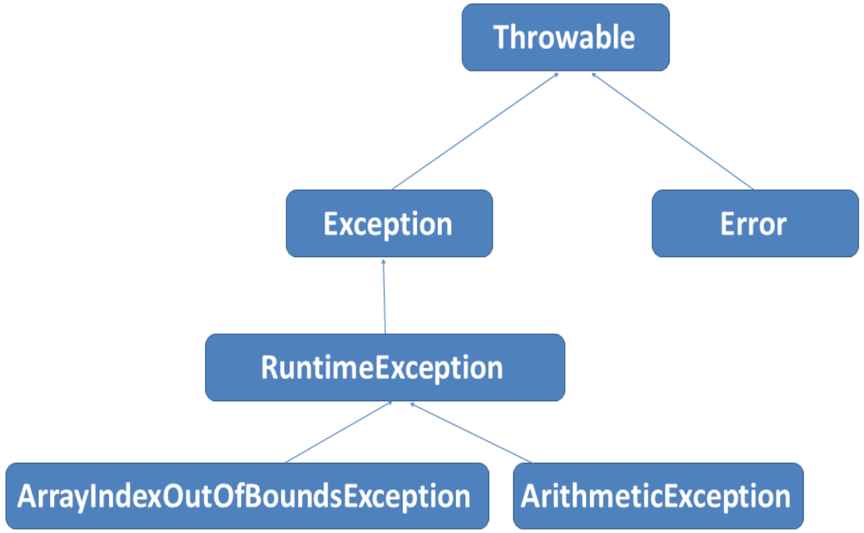
An Exception is-

1. **It is Run Time Error**
2. **It disrupt normal flow of program execution**
3. **If not handled then leads to the crashing of the program**

In case of Exception; JVM will create an object that contains the information about the exception like ExceptionType, line number, method name, class-name, file name etc and that it will throw the same object to the program. The process of creating the object and handing over it to the program is referred to as throwing an exception.

If program is not able to handle the exception then it will be handled by the default handler which is provided by JVM. The default handler terminate the program in the same line where exception was generated and display the stack trace (kind of error log) so that programmer can make correction in program. To display the information printStackTrace() method is used.

Exception Hierarchy



What are Checked & unchecked Exception ?

The try-catch block

```
try{
  statement-1
  statement-2 (code that may generate exception)
  statement-3
}catch(ExceptionType ex){
  code to handle the exception
}
statement-x
```

Exception-pattern-1 (When no exception)
statement-1 -> statement-2 -> statement-3 -> statement-x [No catch block because no exception]

Exception-pattern-2 (When exception occurs)
statement-1 -> statement-2 (Exception occurs) -> catch: code to handle the exception -> statement-x