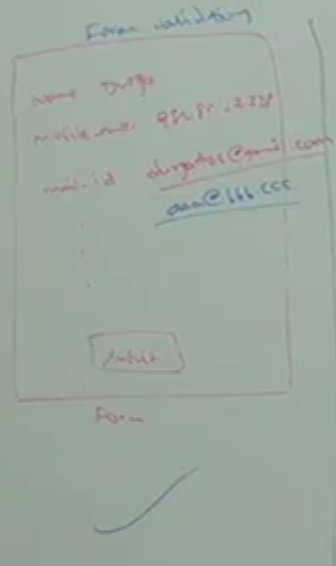
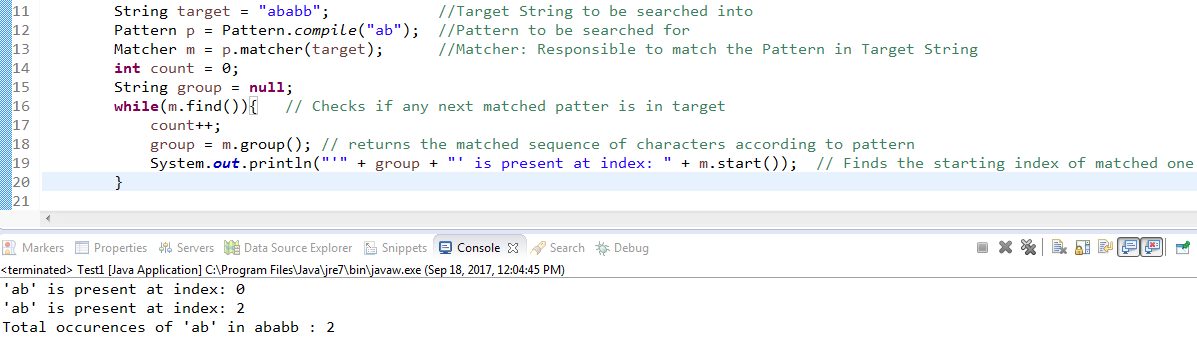
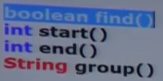
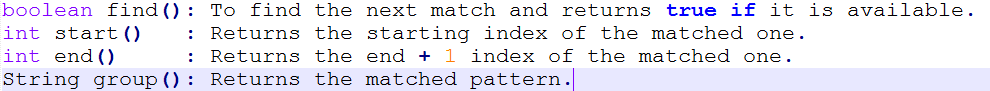
Regular Expression

1. **Definition**:
   1. Represents a group of strings having same pattern. 🡸 Jatin
2. If we want to represent a group of string according to a particular pattern, then we should go for **Regular Expression.**
3. **Dictionary Meaning** 🡺 **Conforming to a standard or pattern 🡸 Jatin**
4. **Example:**
   1. We can write a regular expression to represent all mobile numbers.
   2. We can write a regular expression to represent all email-ids.
5. **Some Application Areas:**
   1. **To Develop Validation Framework  
      Format Validation**   
      Enquiry Form, Registration Form, Login Form  
        
      It just checks format validity of email format not the actual existence of entity 🡺 [aaa.bbb@ccc](mailto:aaa.bbb@ccc) is right according to format of email but in actuality, there is no such mail in existence.
   2. **Pattern Matching Applications**:  
      Find, Copy, Paste, [GREP in UNIX]
   3. **To Develop Translator Like Assemblers, Compilers, Interpreters**Such as Lexical Analysis the 1st step in compilation also know as Tokenization and Scanning.
   4. **To Develop Digital Circuits**
   5. **To Develop Communication Protocols  
      🡺** TCP/IP  
      🡺 UDP
6. d

Let’s Start Programming

1. **Example**:
   1. Take a string 🡺 “abbabbba”
   2. **How many times “ab” is available and where those are available?**
   3. **Answer**: 2 times, 1st 🡺 0th , 2nd 🡺 3rd
   4. Where I want to search this pattern is called 🡺 **Target** **String  
      String target =” abbabbba”**
   5. In the above example, I want to search “ab” so this is the pattern that I’m looking for. With this pattern, create a Pattern object.
      1. **Pattern p = Pattern.compile(“ab”); // Static method in Pattern in java.util.regex**
   6. Now create a Matcher Object
      1. **Matcher matcher = p.matcher(target);**
   7. Matcher object methods
      1. m.find(): boolean: To check whether the next match exists or not
      2. m.start(): starting index: To return the starting index of next match
   8. In the above example, We are using two classes 🡺 Pattern, Matcher in java.util.regex
   9. **Concrete Example**:  
      
   10. **Pattern**: A java object representing compiled version of regular expression. That is java equivalent object of regular expression (Durga said equivalent object of pattern).  
       **How to create Pattern object?**
   11. **Matcher**: Java object to check the match of the pattern in the target String.  
       **How to create?  
       patternObject.matcher()**
   12. **Important methods of Matcher class.**

1. Pattern and Matcher classes came as part of 1.4 and present in java.util.regex