1. Exceptions
Exceptions and Errors are special Java objects families used to describe problems that occurs in applications. They can be split into 3 sub families having a common ancestor (Throwable): Errors, Runtime Exceptions, Checked Exceptions.
1.1. Runtime Exceptions
Runtime Exceptions are problems that happen in your application because you've written bad code. Try this example in Eclipse: <pre class='contentProgramListing' xml:space='preserve'> public static void main(String[] args){ BankAccount ba = null; printAccountDetails(ba); } public static void printAccountDetails(BankAccount ba){ System.out.println(ba.getAccountNumber() + " - " + ba.getBalance()); }</pre> The above example throws a NullPointerException because the getAccountNumber() method is called using a null reference.
1.1.2. Throwing Runtime Exceptions
If some conditions aren't met in your program you may want to throw an Exception. The syntax is easy, simply use the throw keyword and a new instance of a RuntimeException.See the below example:<pre class='contentProgramListing' xml:space='preserve'>public class BankAccount { private String accountNumber; private double balance; // Constructors, getters & setters ... public void debitAccount(double amount) { if(balance < amount){ throw new RuntimeException("Balance too low to execute debit. " + accountNumber); } balance -= amount; } }</pre>
1.1.1 Exercice
Execute the following steps </br> Run a program that throws a NullPointerException (as on the example above). Build a new class that iterates through an array but goes too far Check the Java SE JavaDoc, locate Throwable, Exception and browse through the subclasses of RuntimeException. Note that the notion of subclass will be seen in a later chapter.
1.2. Checked Exceptions
 image test