**Overview**

Apex code is a strongly-typed programming language that executes on the Force.com platform. Using Apex code, you can add business logic to applications, write database triggers, and create Visualforce controllers. Apex code has a tight integration with the database and query language, web services, and email handling support. It also includes features such as asynchronous execution and support for testing.

### Important Reserved Words

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| abstract | Declares a class that contains abstract methods that only have their signature and no body defined. Can also define methods. | ```java
public abstract class Foo {
    protected void method1() { /*… */ }
    abstract Integer abstractMethod();
}
``` |
| break   | Exits the entire loop                           | ```java
while(reader.hasNext()) {
    if (reader.getEventType() == END) {
        break;
    } // process
    reader.next();
}
``` |
| catch   | Identifies a block of code that can handle a particular type of exception | ```java
try {
    // Your code here
} catch (ListException e) {
    // List Exception handling code here
}
``` |
| class   | Defines a class                                 | ```java
private class Foo {
    private Integer x;
    public Integer getX() { return x; }
}
``` |
| continue | Skips to the next iteration of the loop        | ```java
while (checkBoolean) {
    if (condition) {
        continue;
    } // do some work
}
``` |
| do      | Defines a do-while loop that executes repeatedly (at least once) while a Boolean condition remains true | ```java
Integer count = 1;
do {
    System.debug(count);
    count++;
} while (count < 11);
``` |
| else    | Defines the else portion of an if-else statement, that executes if the initial evaluation is untrue | ```java
Integer x, sign;
if (x==0) {
    sign = 0;
} else {
    sign = 1;
}
``` |
| extends | Defines a class or interface that extends another class or interface | ```java
public class MyException extends Exception {} 
``` |
| false   | Identifies an untrue value assigned to a Boolean | ```java
Boolean isNotTrue = false;
``` |
| final   | Defines constants                               | ```java
public class myCls {
    static final Integer intConstant;
}
``` |
| finally | Identifies a block of code that is guaranteed to execute with or without exception | ```java
try {
    // Your code here
} catch (ListException e) {
    // List Exception handling code
} finally {
    // will execute with or without exception
}
``` |
| for     | Defines a loop. The three types of for loops are: iteration over a variable, iteration over a list, and iteration over a query | ```java
for (Integer i = 0, j = 0; i < 10; i++) { System.debug(i+1); } 
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
public enum Season {WINTER, SPRING, SUMMER, FALL};
Season e = Season.WINTER;
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
public class myCls {
static final Integer intConstant;
}
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
for (Account a : [SELECT Id, Name, 
FROM account
WHERE Name LIKE :(s+'%')
]) {
    // Your code
}
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
webService static void makeContact(String lastName) {
    // do some work
}
``` |

---

**Important Reserved Words continued**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| enum    | Defines an enumeration type on a finite set of values | ```java
public enum Season {WINTER, SPRING, SUMMER, FALL};
Season e = Season.WINTER;
``` |
| extends | Defines a class or interface that extends another class or interface | ```java
public class MyException extends Exception {} 
``` |
| finally | Identifies a block of code that is guaranteed to execute with or without exception | ```java
try {
    // Your code here
} catch (ListException e) {
    // List Exception handling code
} finally {
    // will execute with or without exception
}
``` |
| for     | Defines a loop. The three types of for loops are: iteration over a variable, iteration over a list, and iteration over a query | ```java
for (Integer i = 0, j = 0; i < 10; i++) { System.debug(i+1); } 
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
public enum Season {WINTER, SPRING, SUMMER, FALL};
Season e = Season.WINTER;
``` |
| extends | Defines a class or interface that extends another class or interface | ```java
public class MyException extends Exception {} 
``` |
| finally | Identifies a block of code that is guaranteed to execute with or without exception | ```java
try {
    // Your code here
} catch (ListException e) {
    // List Exception handling code
} finally {
    // will execute with or without exception
}
``` |
| for     | Defines a loop. The three types of for loops are: iteration over a variable, iteration over a list, and iteration over a query | ```java
for (Account a : [SELECT Id, Name, 
FROM account
WHERE Name LIKE :(s+'%')
]) {
    // Your code
}
``` |
| global  | Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application. | ```java
webService static void makeContact(String lastName) {
    // do some work
}
``` |
### Important Reserved Words continued

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>if</strong></td>
<td>Defines a condition, used to determine whether a code block should be executed</td>
<td>Integer i = 1; if (i &gt; 0) { // do something; }</td>
</tr>
<tr>
<td><strong>implements</strong></td>
<td>Declares a class or interface that implements an interface</td>
<td>global class CreateTaskEmailExample implements Messaging.InboundEmailHandler { global Messaging.InboundEmailResult handleInboundEmail(Messaging.InboundEmail email, Messaging.InboundEnvelope env) { // do some work, return value; } }</td>
</tr>
<tr>
<td><strong>instanceOf</strong></td>
<td>Verifies at runtime whether an object is actually an instance of a particular class</td>
<td>if (reports.get(0) instanceof CustomReport) { // Can safely cast CustomReport c = (CustomReport) reports.get(0); } else { // Do something with the non-custom-report. }</td>
</tr>
<tr>
<td><strong>interface</strong></td>
<td>Defines a data type with method signatures. Classes implement interfaces. An interface can extend another interface.</td>
<td>public interface PO { public void doWork(); } public class MyPO implements PO { public override doWork() { // actual implementation } }</td>
</tr>
<tr>
<td><strong>new</strong></td>
<td>Creates a new object, sObject, or collection instance</td>
<td>Foo f = new Foo(); MyObject_c mo = new MyObject_c(Name = 'hello'); List&lt;Account&gt; la = new List&lt;Account&gt;();</td>
</tr>
<tr>
<td><strong>null</strong></td>
<td>Identifies a null constant that can be assigned to any variable</td>
<td>Boolean b = null;</td>
</tr>
<tr>
<td><strong>override</strong></td>
<td>Defines a method as overriding another method on a class that’s being extended or implemented.</td>
<td>public virtual class V { public virtual void foo() {/<em>Does nothing</em>/ } } public class RealV implements V { public override void foo() { // Do something real } }</td>
</tr>
<tr>
<td><strong>private</strong></td>
<td>Defines a class, method, or variable that is only known locally, within the section of code in which it is defined</td>
<td>public class OuterClass { // only visible to methods and statements within OuterClass private static final Integer myInt; }</td>
</tr>
<tr>
<td><strong>protected</strong></td>
<td>Defines a method or variable that is visible to any inner classes in the defining Apex class, and to classes that extend the defining class</td>
<td>public class Foo { public void quiteVisible(); protected void lessVisible(); }</td>
</tr>
<tr>
<td><strong>public</strong></td>
<td>Defines a method or variable that can be used by any Apex in this application or namespace</td>
<td>public class OuterClass { public void quiteVisible(); private void almostInvisible(); }</td>
</tr>
<tr>
<td><strong>return</strong></td>
<td>Returns a value from a method</td>
<td>public Integer meaningOfLife() { return 42; }</td>
</tr>
<tr>
<td><strong>static</strong></td>
<td>Defines initialization code, or a method or variable that is initialized only once and is associated with a class</td>
<td>public class OuterClass { // Associated with instance public static final Integer myInt; // Initialization code static { myInt = 10; } }</td>
</tr>
<tr>
<td><strong>super</strong></td>
<td>Invokes a constructor or method on a parent class that is designated as virtual or abstract. Can be used in methods only if they’re designated with the override keyword.</td>
<td>public class AnotherChildClass extends InnerClass { AnotherChildClass(String s) { super(); // different constructor, } }</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>this</td>
<td>Represents the current instance of a class, or calls the previous constructor in a constructor chain</td>
<td>public class Foo { public Foo(String s) { /* ... */ } public Foo() { this('memes repeat'); } }</td>
</tr>
<tr>
<td>throw</td>
<td>Throws an exception, signaling that an error has occurred</td>
<td>public class MyException extends Exception { try { Integer i; if (i &lt; 5) throw new MyException(); } catch (MyException e) { // Your MyException handling // code here }</td>
</tr>
<tr>
<td>transient</td>
<td>Declares instance variables that cannot be saved, and should not be transmitted as part of the view state, in Visualforce controllers and extensions</td>
<td>transient integer currentValue;</td>
</tr>
<tr>
<td>trigger</td>
<td>Defines a trigger on an sObject</td>
<td>trigger MyAccountTrigger on Account (before insert, before update) { if (Trigger.isBefore) { for (Account a : Trigger.old) { if (a.Name == 'okToDelete') { a.addError('You can't delete this record!'); } } } }</td>
</tr>
<tr>
<td>true</td>
<td>Identifies a true value assigned to a Boolean</td>
<td>Boolean mustIterate = true;</td>
</tr>
<tr>
<td>try</td>
<td>In which you can handle an exception if one occurs</td>
<td>try { // Your code here } catch (ListException e) { // List Exception handling code // here }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>webservice</td>
<td>Defines a static method that is exposed as a Web service method that can be called by external client applications. Web service methods must be defined in a global class.</td>
<td>global class MyWebService { webservice static Id makeContact(String lastName, Account a) { Contact c = new Contact( LastName = 'Weissman', AccountId = a.Id); insert c; return c.Id; } }</td>
</tr>
<tr>
<td>while</td>
<td>Executes a block of code repeatedly as long as a particular Boolean condition remains true</td>
<td>Integer count=1; while (count &lt; 11) { System.debug(count); count++; }</td>
</tr>
<tr>
<td>with sharing</td>
<td>Enforces sharing rules that apply to the current user. If absent, code is run under the default system context.</td>
<td>public with sharing class SharingClass { // Code will enforce current user's // sharing rules }</td>
</tr>
<tr>
<td>without sharing</td>
<td>Ensures that the sharing rules of the current user are not enforced</td>
<td>public without sharing class noSharing { // Code won't enforce the current // user's sharing rules }</td>
</tr>
<tr>
<td>virtual</td>
<td>Defines a class or method that allows extension and overrides. You can't override a method with the override keyword unless the class or method has been defined as virtual.</td>
<td>public virtual class MyException extends Exception { // Exception class member // variable public Double d; // Exception class constructor MyException(Double d) { this.d = d; } // Exception class method protected void doIt() () }</td>
</tr>
</tbody>
</table>
### Annotations

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>@future</td>
<td>Denotes methods that are executed asynchronously</td>
<td>global class MyFutureClass {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>@future</td>
</tr>
<tr>
<td></td>
<td></td>
<td>static void myMethod(</td>
</tr>
<tr>
<td></td>
<td></td>
<td>String a, Integer i) {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System.debug(</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Method called with: ' + a +</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' + i);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>// do callout, or execute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>// other long-running code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@isTest</td>
<td>Denotes classes that only contain code used for testing your application.</td>
<td>@isTest private class MyTest {</td>
</tr>
<tr>
<td></td>
<td>These classes don’t count against the total amount of Apex used by your</td>
<td>// Methods for testing</td>
</tr>
<tr>
<td></td>
<td>organization.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@isTest(OnInstall=true)</td>
<td>Denotes a test class or test method that executes on package</td>
<td>@isTest(OnInstall=true) private class TestClass {</td>
</tr>
<tr>
<td></td>
<td>installation</td>
<td></td>
</tr>
<tr>
<td>@isTest(SeeAllData=true)</td>
<td>Denotes a test class or test method that has access to all data in the</td>
<td>@isTest(SeeAllData=true) private class TestClass {</td>
</tr>
<tr>
<td></td>
<td>organization, including pre-existing data that the test didn’t create.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The default is false.</td>
<td></td>
</tr>
<tr>
<td>@deprecated</td>
<td>Denotes methods, classes, exceptions, enums, interfaces, or variables that</td>
<td>@deprecated public void limitedShelfLife() {</td>
</tr>
<tr>
<td></td>
<td>can no longer be referenced in subsequent releases of the managed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>package in which they reside</td>
<td></td>
</tr>
</tbody>
</table>

### Annotations continued

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>@readOnly</td>
<td>Denotes methods that can perform queries unrestricted by the number of</td>
<td>@readOnly private void doQuery() {</td>
</tr>
<tr>
<td></td>
<td>returned rows limit for a request</td>
<td></td>
</tr>
<tr>
<td>@remoteAction</td>
<td>Denotes Apex controller methods that JavaScript code can call from a</td>
<td>@remoteAction global static String getId(String s) {</td>
</tr>
<tr>
<td></td>
<td>Visualforce page via JavaScript remoting. The method must be static and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>either public or global.</td>
<td></td>
</tr>
<tr>
<td>@restResource</td>
<td>Denotes a class that is available as a REST resource. The class must be</td>
<td>@restResource(urlMapping= '/Widget/*') global with sharing class</td>
</tr>
<tr>
<td></td>
<td>global. The urlMapping parameter is your resource’s name and is relative</td>
<td>MyResource() {</td>
</tr>
<tr>
<td></td>
<td>to <a href="https://instance.salesforce.com/services/apexrest/">https://instance.salesforce.com/services/apexrest/</a>.</td>
<td></td>
</tr>
<tr>
<td>@httpGet, @httpPost,</td>
<td>Denotes a REST method in a class annotated with @restResource that the</td>
<td>@httpGet global static MyWidget__c doGet() {</td>
</tr>
<tr>
<td>@httpPatch, @httpPut,</td>
<td>runtime invokes when a client sends an HTTP GET, POST, PATCH, PUT, or</td>
<td></td>
</tr>
<tr>
<td>@httpDelete</td>
<td>DELETE respectively. The methods defined with any of these annotations</td>
<td>@httpPost global static void doPost() {</td>
</tr>
<tr>
<td></td>
<td>must be global and static.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>@httpDelete global static void doDelete() {</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Primitive Types

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blob</td>
<td>Binary data stored as a single object</td>
<td>Blob myBlob = Blob.valueOf('idea');</td>
</tr>
<tr>
<td>Boolean</td>
<td>Value that can only be assigned true, false, or null</td>
<td>Boolean isWinner = true;</td>
</tr>
<tr>
<td>Date</td>
<td>Particular day</td>
<td>Date myDate = Date.today(); Date weekStart = myDate.toStartOfWeek();</td>
</tr>
<tr>
<td>DateTime</td>
<td>Particular day and time</td>
<td>DateTime myDateTime = DateTime.now(); DateTime newDateTime = myDateTime.addMonths(2);</td>
</tr>
<tr>
<td>Decimal</td>
<td>Number that includes a decimal point. Decimal is an arbitrary precision number.</td>
<td>Decimal myDecimal = 12.4567; Decimal divDec = myDecimal.divide(7, 2, System.RoundingMode.UP); System.assertEquals(divDec, 1.78);</td>
</tr>
<tr>
<td>Double</td>
<td>64-bit number that includes a decimal point. Minimum value (-2^{63}). Maximum value of (2^{63}-1)</td>
<td>Double d=3.14159;</td>
</tr>
<tr>
<td>ID</td>
<td>18-character Force.com record identifier</td>
<td>ID id='00300000003T2PGAA0';</td>
</tr>
<tr>
<td>Integer</td>
<td>32-bit number that doesn’t include a decimal point. Minimum value (-2,147,483,648 ) — maximum value of (2,147,483,647)</td>
<td>Integer i = 1;</td>
</tr>
<tr>
<td>Long</td>
<td>64-bit number that doesn’t include a decimal point. Minimum value (-2^{63}) — maximum value of (2^{63}-1).</td>
<td>Long l = 2147483648L;</td>
</tr>
<tr>
<td>Object</td>
<td>Any data type that is supported in Apex</td>
<td>Object obj = 10; // Cast the object to an integer. Integer i = (Integer)obj; System.assertEquals(i, i); Object obj = new MyApexClass(); // Cast the object to the // MyApexClass custom type. MyApexClass mc = (MyApexClass) obj; // Access a method on the // user-defined class. mc.someClassMethod();</td>
</tr>
</tbody>
</table>

### Primitive Types continued

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Set of characters surrounded by single quotes</td>
<td>String s = 'repeating memes';</td>
</tr>
<tr>
<td>Object</td>
<td>Any data type that is supported in Apex</td>
<td>String s = 'repeating memes';</td>
</tr>
<tr>
<td>Time</td>
<td>Particular time</td>
<td>Object obj = 10; // Cast the object to an integer. Integer i = (Integer)obj; System.assertEquals(i, i); Object obj = new MyApexClass(); // Cast the object to the // MyApexClass custom type. MyApexClass mc = (MyApexClass) obj; // Access a method on the // user-defined class. mc.someClassMethod();</td>
</tr>
</tbody>
</table>

### Trigger Context Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>isExecuting</td>
<td>Returns true if the current context for the Apex code is a trigger only</td>
</tr>
<tr>
<td>isInsert</td>
<td>Returns true if this trigger was fired due to an insert operation</td>
</tr>
<tr>
<td>isUpdate</td>
<td>Returns true if this trigger was fired due to an update operation</td>
</tr>
<tr>
<td>isDelete</td>
<td>Returns true if this trigger was fired due to a delete operation</td>
</tr>
<tr>
<td>isBefore</td>
<td>Returns true if this trigger was fired before any record was saved</td>
</tr>
<tr>
<td>isAfter</td>
<td>Returns true if this trigger was fired after all records were saved</td>
</tr>
<tr>
<td>isUndelete</td>
<td>Returns true if this trigger was fired after a record was recovered from the Recycle Bin</td>
</tr>
<tr>
<td>new</td>
<td>Returns a list of the new versions of the sObject records. (This sObject list is available only in insert and update triggers. The included records can be modified only in before triggers.)</td>
</tr>
<tr>
<td>newMap</td>
<td>A map of IDs to the new versions of the sObject records. (Only available in update and delete triggers.)</td>
</tr>
<tr>
<td>old</td>
<td>Returns a list of the old versions of the sObject records. (Only available in update and delete triggers.)</td>
</tr>
<tr>
<td>oldMap</td>
<td>A map of IDs to the old versions of the sObject records. (Only available in update and delete triggers.)</td>
</tr>
<tr>
<td>size</td>
<td>The total number of records in a trigger invocation, both old and new.</td>
</tr>
</tbody>
</table>
### Standard Interfaces (Subset)

**Database.Batchable**

```java
global (Database.QueryLocator | Iterable<sObject>)
    start(Database.BatchableContext bc) {}
    global void execute(Database.BatchableContext BC, list<P>){}
    global void finish(Database.BatchableContext BC){}
```

**Schedulable**

```java
global void execute(ScheduleableContext SC) {}
```

**Messaging.InboundEmailHandler**

```java
global Messaging.InboundEmailResult handleInboundEmail(Messaging.inboundEmail email, Messaging.InboundEnvelope env){}
```

**Comparable**

```java
global Integer compareTo(Object compareTo) {}
```

### Collection Types

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| List       | Ordered collection of typed primitives, sObjects, objects, or collections that are distinguished by their indices | `// Create an empty list of String
List<String> myList = new List<String>();
myList.add('hi');
String x = myList.get(0);` |
| Map        | Collection of key-value pairs where each unique key maps to a single value. A key can be any primitive data type, while a value can be a primitive, an sObject, a collection type, or an object. | `Map<String, String> MyStrings = new Map<String, String>{
    'a' => 'b',
    'c' =>
        'd'.toUpperCase();
    Account myAcct = new Account();
    Map<Integer, Account> m = new Map<Integer, Account>();
    m.put(1, myAcct);` |
| Set        | Unordered collection that doesn't contain any duplicate elements. | `Set<Integer> s = new Set<Integer>();
s.add(12);
s.add(12);
System.assert(s.size()==1);` |

### Apex Data Manipulation Language (DML) Operations

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| insert     | Adds one or more records | `Lead l = new Lead(Company='ABC', LastName='Smith');
insert l;` |
| delete     | Deletes one or more records | `Account[] webAccts = [SELECT Id, Name
FROM Account WHERE Name = 'DotCom';
try { delete webAccts;
} catch (DmlException e) { // Process exception here });` |
| merge      | Merges up to three records of the same type into one of the records, deleting the others, and re-parenting any related records | `List<Account> ls = new List<Account>{
    new Account(Name='Acme Inc.'),
    new Account(Name='Acme');
} insert ls;
Account masterAcct = [SELECT Id, Name
FROM Account WHERE Name = 'Acme Inc.'
LIMIT 1];
Account mergeAcct = [SELECT Id, Name FROM
Account WHERE Name = 'Acme'
LIMIT 1];
try { merge masterAcct mergeAcct;
} catch (DmlException e) {` |
| undelete   | Restores one or more records from the Recycle Bin | `Account[] savedAccts = [SELECT Id, Name
FROM Account WHERE Name = 'Trump'
ALL ROWS];
try { undelete savedAccts;
} catch (DmlException e) {` |
| update     | Modifies one or more existing records | `Account a = new Account(Name='Acme2');
insert(a);
Account myAcct = [SELECT Id, Name,
    BillingCity FROM Account WHERE
Name = 'Acme2' LIMIT 1];
myAcct.BillingCity = 'San Francisco';
try {
    update myAcct;
} catch (DmlException e) {` |
| upsert     | Creates new records and updates sObject records within a single statement, using a specified field to determine the presence of existing objects, or the ID field if no field is specified. | `Account[] acctsList = [SELECT Id, Name,
    BillingCity FROM Account WHERE
BillingCity = 'Bombay'];
for (Account a : acctsList) {
    (a.BillingCity = 'Mumbai');
Account newAcct = new Account( Name = 'Acme', BillingCity = 'San Francisco');
acctsList.add(newAcct);
try { upsert acctsList;
} catch (DmlException e) {` |
Commonly Used Methods

**System Class**
- abortJob
- assertEquals
- currentPageReference
- debug
equals
- ReadWriteMode
- isBatch
- isScheduled
- purgeOldAsyncJobs
- resetPassword
- schedule
- setPassword
- today

**Math Class**
- abs
- acos
- asin
- atan
- atan2
- cbrt
- ceil
- cos
- cosh
- exp
- floor
- log
- log10
- max
- min
- mod
- pow
- random
- rint
- round
- roundToLong
- signum
- sin
- sinh
- sqrt
- tan
- tanh

**DescribeSObjectResult Class**
- fields
- getChildRelationships
- getKeyPrefix
- getLocalName
- getRecordTypeInfos
- getRecordTypeInfosByID
- isAccessible
- isCustom
- isDependentPicklist
- isExternalID
- isGroupable
- isIdLookup
- isNameField
- isNamePointing
- isPermissionable
- isRestrictedDelete
- isRestrictedPicklist
- isSortable
- isUnique
- isWriteRequiresMasterRead
- getByteLength
- getController
- getDefaultValue
- getDefaultValueFormula
- getInlineHelpText
- getName
- getNumberOfValues
- getPicklistValues
- getRecordTypeInfosByName
- getRelationshipName
- getRelationshipOrderByField
- getScale
- isAccessControl
- isAutoNumber
- isCalculated
- isCascadeDelete
- isCaseSensitive
- isCreateable
- isCustom
- isDefaultedOnCreate
- isDependentPicklist
- isExternalID
- isFilterable
- isGroupable
- isHtmlFormatted
- isIdLookup
- isNameField
- isNamePointing
- isPermissionable
- isRestrictedDelete
- isRestrictedPicklist
- isSortable
- isUnique
- isWriteRequiresMasterRead

**DescribeFieldResult Class**
- getByteLength
- getController
- getDefaultValue
- getInlineHelpText
- getName
- getNumberOfValues
- getPicklistValues
- getRecordTypeInfosByName
- getScale
- isAccessControl
- isAutoNumber
- isCalculated
- isCascadeDelete
- isCaseSensitive
- isCreateable
- isCustom
- isDefaultedOnCreate
- isDependentPicklist
- isExternalID
- isFilterable
- isGroupable
- isHtmlFormatted
- isIdLookup
- isNameField
- isNamePointing
- isPermissionable
- isRestrictedDelete
- isRestrictedPicklist
- isSortable
- isUnique
- isWriteRequiresMasterRead

**Limits Class**
- getAggregateQueries
- getAsyncCalls
- getCallouts
- getCpuTime
- getDMLRows
- getDMLStatements
- getEmailInvocations
- getFutureCalls
- getHeapSize
- getMobilePushApexCalls
- getQueryLocatorRows
- getQueryRows
- getQueueableJobs
- getSoslQueries

**UserInfo Class**
- getDefaultCurrency
- getLanguage
- getLocale
- getOrganizationId
- getProfileId
- getTimeZone
- getUiTheme
- getUiThemeDisplayed
- getUserEmail
- getUserId
- getUserRole
- isCurrentUserLicensed
- isMultiCurrencyOrganization

---

```
Commonly Used Methods continued

**DescribeSObjectResult Class**
- getByteLength
- getController
- getDefaultValueFormula
- getInlineHelpText
- getName
- getNumberOfValues
- getPicklistValues
- getRecordTypeInfosByName
- getScale
- isAccessControl
- isAutoNumber
- isCalculated
- isCascadeDelete
- isCaseSensitive
- isCreateable
- isCustom
- isDefaultedOnCreate
- isDependentPicklist
- isExternalID
- isFilterable
- isGroupable
- isHtmlFormatted
- isIdLookup
- isNameField
- isNamePointing
- isPermissionable
- isRestrictedDelete
- isRestrictedPicklist
- isSortable
- isUnique
- isWriteRequiresMasterRead

**Limits Class**
- getAggregateQueries
- getAsyncCalls
- getCallouts
- getCpuTime
- getDMLRows
- getDMLStatements
- getEmailInvocations
- getFutureCalls
- getHeapSize
- getMobilePushApexCalls
- getQueryLocatorRows
- getQueryRows
- getQueueableJobs
- getSoslQueries

**UserInfo Class**
- getDefaultCurrency
- getLanguage
- getLocale
- getOrganizationId
- getProfileId
- getTimeZone
- getUiTheme
- getUiThemeDisplayed
- getUserEmail
- getUserId
- getUserRole
- isCurrentUserLicensed
- isMultiCurrencyOrganization
```
For other cheatsheets:
http://developer.salesforce.com/cheatsheets