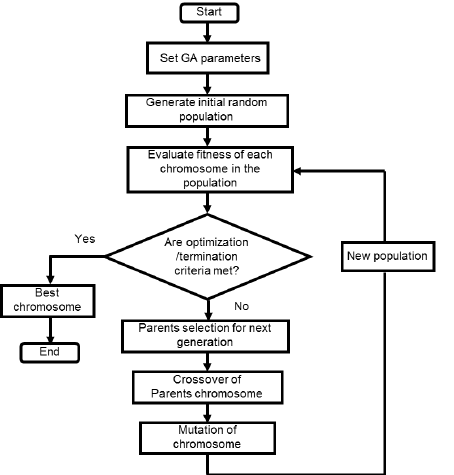
Some basic terms related to Genetic algorithms:

* **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
* **Chromosomes** − A chromosome is one such solution to the given problem.
* **Gene** − A gene is one element position of a chromosome.
* **Allele** − It is the value a gene takes for a particular chromosome.
* **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
* **Phenotype** − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.
* **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.
* **Fitness Function** − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.
* **Genetic Operators** − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.



**Applet**

An **applet** is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

An applet is a Java class that extends the java.applet.Applet class.

A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.

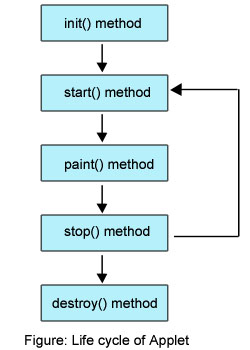
The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.

**Life Cycle of an Applet**

Four methods in the Applet class gives you the framework on which you build any serious applet −

* **init** − This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
* **start** − This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
* **stop** − This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
* **destroy** − This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
* **paint** − Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

These methods are known as Applet Life Cycle methods. These methods are defined in java.applet.Applet class except paint() method. The paint() method is defined in java.awt.Component class, an indirect super class of Applet.

[](http://way2java.com/wp-content/uploads/2011/01/Life-cycle-of-Applet.jpg)