ABSTRACTION

A representation is something that stands for and takes the place of something else.

Example
The concept of the number five is represented by the numeral 5. Other representations of five include the Roman numeral V and the binary value 101.

An abstraction represents some thing in general terms and ignores its many concrete qualities.

Example
This sketch is easily recognized as that of a bunny rabbit.

Hardware engineers use abstraction by creating simplified representations that hide the complex mechanisms and processes inside the device.

Example
A good example of engineering abstraction is the braking system on your automobile, a system with many parts that is represented to the driver as a simple foot pedal.

Source: http://www.familycar.com/brakes.htm
Abstraction is important to engineering for three reasons.

- It makes devices easier to use.

**Example**
An automobile driver stops his or her car by stomping the brake pedal.

- It makes devices easier to maintain.

**Example**
An automobile mechanic can change the pads on the brakes without touching any other part of the system.

- It allows for the same user representation to have different implementations.

**Example**
One automobile has disc brakes, another has anti-lock brakes. In both systems the driver stops the car by stomping the pedal.

**Data Abstraction**
Computer programs use two kinds of abstraction to hide the inner workings of the physical machine on which it executes. The first is *data abstraction*, which involves the separation of the essential characteristics of data from the physical details of how it is implemented.

**Example**
Consider the program statement:

```plaintext
a = -5;
```

The symbol \(-5\) represents the integer \(-5\). This is a data abstraction because the circuitry of the physical computer stores signed integers in a binary form called *two’s complement*. The two’s complement form of \(-5\) is \(111111111111011\).

The variable \(a\) is another data abstraction, representing the location in the computer's memory device where the value \(111111111111011\) is stored.
A **class** is a mechanism whereby a programmer can create his or her own data abstractions.

**Example**
The Java language provides a `GregorianCalendar` class from which you can build an object containing a specific instance of time in the Gregorian calendar. This is done with the Java statement:

```java
GregorianCalendar cal = new GregorianCalendar();
```

The calendar object stores the time as the number of milliseconds since midnight of January 1, 1970 (Greenwich Mean Time). This allows the object to keep time with great precision, but it is not readily understood by humans. So the `GregorianCalendar` class provides methods that you can call to view the date in terms of month, day, year, hours, minutes, etc.

For instance, suppose the `GregorianCalendar` object `cal` above contains the value 3600000 (i.e. 3,600 seconds past midnight of January 1, 1970 GMT). This Java statement:

```java
System.out.print( cal.getTime().toString() );
```

Prints the correct date and time:

```
Thu Jan 01 01:00:00 GMT 1970
```

**Procedural Abstraction**
When a sequence of operations has a well-defined effect, then a human being can combine those operations into a single “chunk” of knowledge, represented as a single task. This process is called **procedural abstraction**.

**Example**
The length of the hypotenuse $t$ of a right triangle is $t = \sqrt{b^2 + h^2}$ where $b$ is the base length and $h$ is its height. The Java statement below does this calculation:

```java
    t = sqrt( pow(b,2) + pow(h,2) );
```

`sqrt` and `pow` are procedural abstractions that calculate square roots and powers while hiding the details of the calculations.
A *method* is a mechanism whereby a programmer can create his or her own procedural abstractions.

**Example**
The following method calculates the length of the hypotenuse of a right triangle when given the base length $b$ and height $h$.

```java
double hypotenuse( double a, double b )
{
    return Math.sqrt( Math.pow( a, 2 ) + Math.pow( b, 2 ) );
}
```

Having defined this method, the programmer can calculate the hypotenuse of several right triangles by calling it with different data:

```java
t1 = hypotenuse( 3.0, 5.0 );
t2 = hypotenuse( 2.0, 2.0 );
```