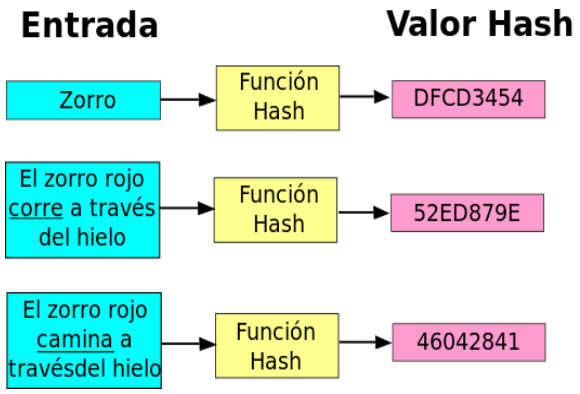
Programación II
Hash

Una función hash H es una función computable mediante un algoritmo, que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

$H: U \rightarrow M$
$x \rightarrow h(x)$,

Al conjunto U se le llama dominio de la función hash. A un elemento de U se le llama **preimagen** o dependiendo del contexto **clave** o **mensaje**.

Al conjunto M se le llama imagen de la función hash. A un elemento de M se le llama **valor hash**, **código hash** o simplemente **hash**.



Normalmente el conjunto M tiene un número elevado de elementos y U es un conjunto de cadenas con un número más o menos pequeño de símbolos. Por esto se dice que estas funciones resumen datos del conjunto dominio.

La idea básica de un valor hash es que sirva como una representación compacta de la cadena de entrada. Por esta razón decimos que estas funciones **resumen** datos del conjunto dominio.

Ejemplo

```
public class Tad1 {
      public static int cont;

      public Tad1(){    cont++;     }

       @Override
      public int hashCode() {
              return (cont);
      }
}

public class test {
      public static void main(String[] args) {
            Tad1 a  = new Tad1();
            Tad1 b  = new Tad1();

            System.out.println(a. hashCode());
      }

}
```

Colisión

Se dice que se produce una **colisión** cuando dos entradas distintas de la función de hash producen la misma salida. De la definición de función hash podemos decir que U, el dominio de la función, puede tener infinitos elementos. Sin embargo M, el rango de la función, tiene un número finito de elementos debido a que el tamaño de sus cadenas es fijo. Por tanto la posibilidad de existencia de colisiones es intrínseca a la definición de función hash. Una buena función de hash es una que tiene pocas colisiones en el conjunto esperado de entrada. Es decir, se desea que la probabilidad de colisión sea muy baja.

Se dice que la función hash es inyectiva cuando cada dato de entrada se mapea a un valor hash diferente. En este caso se dice que la función hash es perfecta. Para que se dé, es necesario que la cardinalidad del conjunto dominio sea inferior o igual a la cardinalidad del conjunto imagen. Normalmente sólo se dan funciones hash perfectas cuando las entradas están preestablecidas.

Ejemplo:Mapear los días del año en números del 1 al 366 según el orden de aparición.

Formalización:

$k1 \neq k2$ implica $h(k1) \neq h(k2)$

Cuando no se cumple la propiedad de inyectividad se dice que hay **colisiones**.

Hay una **colisión** cuando $k1 \neq k2$ y $h(k1) = h(k2)$

```
public class Persona {
	public int dni;

	public Persona (int dni){	dni = dni; }

	 @Override
	public int hashCode() {
		return (dni);
	}
}
```

¿Cuando hay colisiones en el TAD Persona?

Ejercicio

Hacer el TAD generala que modele la tirada de 6 dados e implemente una función de hash a partir de los valores de los dados.

Hacer que la función no tenga colisiones.