Turing - Java Dictionary
A Concise Guide for Translating Turing Constructs and Routines into Java

The following tables are a *quick reference comparison* of the basic elements of Java, and Turing. This guide is designed for those who are familiar with Turing (or OOT) and are beginning to learn programming in Java. All commands below are those that would be used in writing code for Java *applications*.

In the tables below, the object '**c**' must be an object instantiated from the *Console* class of Holt Software Associates. A statement that imports the *hsa* class library must be included in the program. When using standard input and output in Java, one must import the *java.io* class library. Also, the *java.awt* class library should be included whenever colour or graphics are used.

Variable Declarations

Language	**Turing**	**Java**
Declaring an Integer Variable	var age : int	int age;
Declaring a Real Variable	var mass : real	double mass;
Declaring a String	var name : string	String name;

Declaring and Initializing a Variable at the same time	var age : int := 35
	
	var word : string := "Joe"
		int age = 35;
		
		String word = new String("Joe");
		or
		String word = "Joe";

Declaring and Setting the Value of a Constant	const pi : real := 3.14159	final double pi = 3.14159;

Declaring an Array	var agelist : array 1..5 of int	int[ ] ageList;
		ageList = new int[5];
		or
		int[ ] ageList = new int[5]

Declaring a 2-Dimensional Array	var paygrid : array 1..5,1..7 of int	int[ ][ ] table;
		table = new int[5][7];

Standard Input and Output

Language	**Turing**	**Java**

Outputting to Screen	put "Hello"	system.out.println ("Hello");
		or
		c.println ("Hello");

Outputting to Screen without Advancing the Cursor	put "Bye now" ..	c.print ("Bye now");
Outputting into a Fixed-Length Field on the Screen	put "TITLE" : 12	c.println ("TITLE", 12);
Outputting into a Fixed-Length Field with # of Decimal Places Specified	put average : 10 : 2	c.println (average,10,2);
Outputting Several Items on the Screen in a Single Statement	put "You are ", age , "yrs old."	c.println ("You are " + age + "yrs old");
Inputting a Single Token from the Keyboard and Storing in a String Variable	get firstword	firstword = c.readString( );
Inputting an Entire Line of Data from the Keyboard and Storing in a String Variable	get fullname : *	fullname = c.readLine( );
Inputting an Integer from the Keyboard	get age	age = c.readInt( );
Inputting a Real Number from the Keyboard	get height	height = c.readDouble( );

Basic Operators

	Turing		Java	
	Operator	**Example**	**Operator**	**Example**
Assignment Operator	**:=**	mark := 93	**=**	mark = 3;
Equality Comparison Operator	**=**	(age = 16)	**==**	(age == 16)
Real Division	**/**	16 / 5	**/**	(double)16 / 5 *or* 16.0 / 5 (where at least one of the operands is a real number)
Integer Division : Quotient	**div**	16 div 5	**/**	16 / 5
Integer Division : Remainder	**rem**	16 rem 5	**%**	16 % 5
Exponentiation	**	2**8	*none*	no built-in operator
not	**not** *or* **~**	age not= 16 age ~= 16	**!**	age != 16

and	**and**
	or
	&
		age=16 and count=10
		age=16 & count=10
			&&	age==16 && count==10

or	**or** *or* **|**	(age<16) or (age>85) (age<16) | (age>85)	**||**	(age<16) || (age>85)

Miscellaneous Commands

Language	**Turing**	**Java**
Set the Colour of Text	colour (2)	c.setTextColour (Color.green);
Set the Background Colour	colourback (4)	c.setTextBackgroundColour (Color.red);
Clear the Screen	cls	c.clear( );
Move the Cursor to a Specific Location on the Screen	locate(10, 20)	c.setCursor (10,20);
Store a Randomly Chosen Integer in a Variable	randint (numb,1,50)	numb = (int) (Math.random( )*50) + 1;
Output the Length of a String to the Screen	put length (name)	c.println (name.length( ));

Store in an Integer Variable the Result of a Real Number Rounded to the Nearest Integer	var numb : int
	numb := round(1.55)
		int numb;
		numb = (int) Math.round(1.55) ;

Display the ASCII Number that Corresponds to a Keyboard Character	put ord("A")	c.println((int) 'A'); *(Note that 'A' is of type* ***char****, not String. Also, in Java, the unicode number is displayed. This is the same as the ASCII number since ASCII is a subset of unicode.)*
Display the Keyboard Character that Corresponds to an ASCII Number	put chr(65)	c.println((char)65);

Various Graphics Commands	
	drawbox (50,50,70,70,4)
	
	drawfillbox (50,50,70,70,4)
	drawoval (100,100,20,20,4)
	drawline (20,30,60,60,4)
	drawarc (90,90,40,30,0,90,4)
	drawmapleleaf(40,50,60,70,4)
	drawstar(40,50,60,70,4)
	Draw.Star(40,50,60,70,4)
		c.setColour (Color.red);
		c.drawRect (50,50,70,70);
		
		c.fillRect (50,50,70,70);
		c.drawOval (100,100,20,20);
		c.drawLine(20,30,60,60);
		c.drawArc(90,90,40,30,0,90);
		c.drawMapleLeaf(40,50,60,70);
		c.drawStar(40,50,60,70);
		c.drawStar(40,50,60,70);

Control Structures

Language	**Turing**	**Java**

Selection Structure (simple)	if (age<16) then
	      put "Too young to drive."
	end if
		if (age<16)
		{
		      c.println ("Too young to drive.");
		}

Selection Structure (2-way)	if (age<16) then
	     put "Too young to drive."
	else
	     put "Old enough!"
	end if
		if (age<16)
		{
		      c.println ("Too young to drive.");
		}
		else
		{
		       c.println ("Old enough!);
		}

Selection Structure (compound)	if (age<16) then
	      put "Too young to drive."
	elsif (age >= 80) then
	     put "Driver test req'd."
	else
	     put "Standard driving age."
	end if
		if (age<16)
		{
		      c.println ("Too young to drive.");
		}
		else if (age >=80)
		{
		      c.println ("Driver test req'd.");
		}
		else
		{
		      c.println ("Standard driving age.");
		}

Case Construct (also called Switch Construct)	put "Enter mark out of 10: "
	get mark
	
	case mark of
	      label 9,10:
	             put "Great"
	      label 7,8:
	             put "Good"
	       label 6:
	             put "Fair"
	       label:
	             put "Poor"
	end case
		c.println ("Enter mark out of 10: ");
		mark = c.readInt( );
		
		switch (mark)
		{
		       case 9: case 10:
		              c.println ("Great");
		              break;
		       case 7: case 8:
		              c.println ("Good");
		              break;
		       case 6:
		              c.println ("Fair");
		              break;
		       default:
		              c.println ("Poor");
		              break;
		}

Counted Loop	for i : 1..12
	    put "Hi ! "
	end for
		for (int i=0 ; i <12 ; i++)
		{
		    c.println ("Hi !");
		}

Counted Loop (loop index incremented by 2)	for i : 1..50 by 2
	    put i
	end for
		for (int i=1 ; i <= 50 ; i=i+2)
		{
		    c.println (i);
		}

Counted Loop (descending loop index)	for descending i : 5..1
	    put "countdown " , i
	end for
	put "blastoff!!"
		for (int i=5 ; i > 0 ; i--)
		{
		      c.println ("countdown" + i);
		}

Conditional Loop	sum := 0
	mark := 0
	loop
	    exit when mark < 0
	    sum := sum + mark
	    put "Enter mark:"
	    get mark
	end loop
		sum =0;
		mark =0;
		while (mark >= 0)
		{
		    sum=sum+mark;
		    c.println ("Enter mark:");
		    mark = c.readDouble( );
		}
		or
		sum = 0;
		mark = 0;
		do
		{
		    sum=sum+mark;
		    c.println ("Enter mark:");
		    mark = c.readDouble( );
		}
		while (mark >= 0);

Conditional Loop with an Exit in the Loop Body	sum := 0
	mark := 0
	loop
	    put "Enter mark:"
	    get mark
	    exit when mark < 0
	    sum := sum + mark
	end loop
		sum = 0;
		mark = 0;
		while (true)
		{
		    c.println ("Enter mark:");
		    mark = c.readDouble( );
		    if (mark < 0)
		    {
		        break;
		    }
		    sum = sum + mark;
		}

Infinite Loop	loop
	    put "Hi!!!!!!!"
	end loop
		while (true)
		{
		      c.println ("Hi!!!!!!!");
		}
		or
		for (; ;)
		{
		     c.println ("Hi!!!!!!!");
		}

Subprogram / Method Definitions

Language	**Turing**	**Java**

Function	function *triple* (num : real) : real
	      result 3.0 * num
	end triple
		public static double *triple* (double num)
		{
		     return 3.0*num;
		}

	In the above examples, the function name appears in italics, and the function type appears underlined.	

Procedure	Procedure *greet* (name : string)
	      put "Hello ", name
	end greet
		static public void *greet* (String name)
		{
		      c.println ("Hello" + name);
		}

	In the above examples, the procedure name appears in italics. Note that in Java, procedures take the form of a function that has type "void" (no return or result statement is used).	

A Delay Command

Language	**Turing**	**Java**

Cause the Program to Pause or Delay for Approximately One Second Before Continuing	delay(1000)	*We create a user-defined method that will work in the same way as Turing's delay command. The method definition should be located below and outside the main method, but inside the class.*
		
		`public static void delay (int timeUnit)`
		`{`
		`    int adjVar = 100000;`
		`     for(int i=0; i<adjVar; i++)`
		`     {`
		`          for( int j=0; j< timeUnit; j++)`
		`          {`
		`               double junkvar;`
		`               junkvar = Math.PI*Math.PI;`
		`          }`
		`     }`
		`}`
		
		*Note that the timing of the first version shown above is dependent upon the processor speed of the computer running the program. By trial and error, the value of the local variable **adjVar** can be altered so that using 1000 as the argument to this subprogram results in a 1 second delay.*
		
		Below is an alternate definition of the delay subprogram that is not dependent on processor speed.
		
		`public static void delay(int ms)`
		`{`
		`        try`
		`        {`
		`             Thread.sleep (ms);`
		`        }`
		`        catch (Exception e)`
		`        {`
		`                ;`
		`        }`
		`}`
		
		Once one of these definitions of the delay command are placed into the class, it can be called as follows from within any other method in the class (including main):
		
		delay(1000);

Concurrency

Note: Concurrency refers to the process of dividing the flow of the program into two or more separate branches that execute simultaneously. The chart shown below only gives the most simple example of concurrency. In Java, a proper understanding of how concurrency works requires the understanding of some fairly advanced concepts not addressed here. Both programs include some explanatory comments embedded in the code. The two Java classes must be located in different files, but in the same folder.

Language	**Turing**	**Java**

```
%   a "process" is defined in the structure
%  below

process ByeProcess
     for i:1..100
         put"GOODBYE"
     end for
end ByeProcess


% when the keyword fork is used,  the
%  process defined above is  started while
% the program simultaneously continues
% executing the remaining lines of code

fork ByeProcess


for k:1..100
    put"HELLO"
end for
```

```
// the instance class shown below defines a "process" or
// "Thread" that can be started from an application class

public class ByeThread extends Thread
{

    public ByeThread( )
    {
         super( );
    }

    public void run ( )
    {
         for (int i = 0 ; i < 100 ; i++)
         {
              System.out.println ("GOODBYE");
         }
    }

} // end of ByeThread class


// below is an application class that starts the Thread
// defined above from inside of its main method

import java.awt.*;

public class ThreadTester
{

    public static void main (String[] args)
    {

        ByeThread myThread;
        myThread = new ByeThread ();
        myThread.start ();

        for (int i = 0 ; i < 100 ; i++)
        {
           System.out.println ("HELLO");
        }

    } // end of main method


} // end of ThreadTest class
```