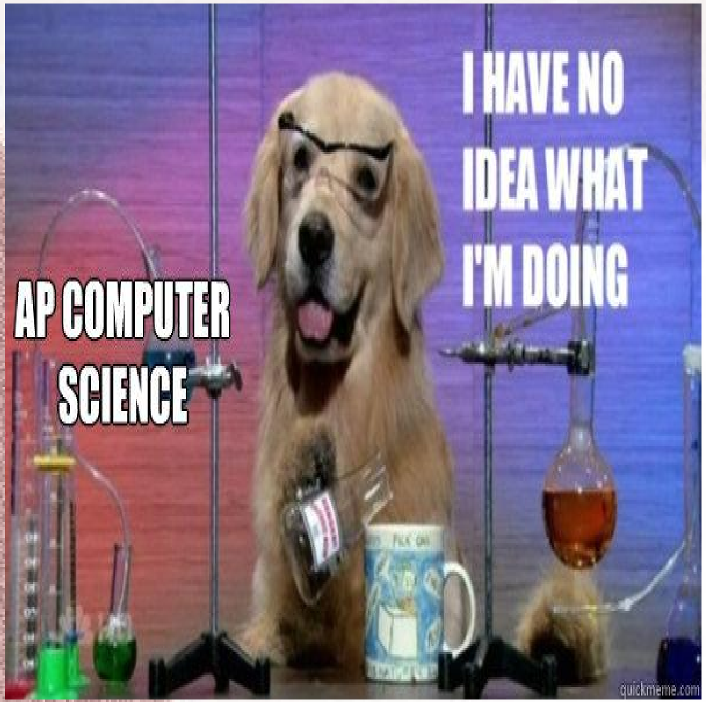
Chapter 1 – Introduction to java



History of java

- Java was created by by Sun Microsystems team led by James Gosling (1991)
- It was mainly used for home appliance, it later became a general purpose programming language

Type of programs in java

- There are two type of programs :
- **applets** : “internet application”, mostly for web browser
- -Have to use GUI (graphic user interface)

  applications : It must contains a main method

  -can use graphics, GUI , or console I/O (input/output)

Structure of a java programs

Single Line

Multiple Lines

```
//This is a comment
*This is also a comment*
/** this is a Javac comment /
public class FirstProgram {


}
```

Mostly use the first 2 to comment codes

Header

Body

Structure of a java programs

```
public class FirstProgram {

  public static void main(String [] args) {


  }
}
```

Method body

Method header

Example of programs

```
public class HelloWorld {
   public static void main(String [] args){
     System.out.println(“Hello world”);
  }
}
```

Output

```
Hello world
```

Compile vs run-time vs logic errors

- Compile-time
- -if can't compile, then is compile-time
- -mostly syntax, ie: adding string with int
- Run-time
- - Error occur during the execution of
-    program(most difficule to find),
   ie: division by zero
- Logic
- -error in the algorithm
- Example: calculation mistakes
-

Example of compile/run-time errors

Compile time error
System.out.println(Hello darkness my old friend);
Output: Program can't compile because no quotation

Run-time errors

Int ten = 10;
Int zero = 0;
System.out.println(“10 divide by 0”);
System.out.println(ten/zero);

Output: 10 divide by 0 ****PROGRAM CRASHED!!!!!****

System.out.println

- Java use object to perform “action”
- -system.out : is used to send output to screen
- -println : is used to print the “object” into the screen

Terminology

- **Bug** : It is when the programs have errors in it, the process of eliminating bug is call “Debugging”
-
- **Syntax error**: a grammar mistake such as mispronouncing certain words. The compile can spot these mistakes

Identifiers

- We can think of identifiers as variables to name a data or an item (ie: class , method , object, etc...)
- Rule of thumbs for naming them is to make it short and simple.

*note: Do not put random identifiers such as pies, you will get the meaning of the

Identifiers

- **Rules for identifiers:**
  -It can have : 1) Letters
  2) Digits
  3) Character underscore (_)
  4) Dollar sign ($)
- -It cannot start with a **Digits**
- -Cannot be a reserve word (ie: public, super, this, if , for, etc...)
- - There is not limit for the length
- -Java is case sensitive, So Rate, rate, _rate are all different identifier

Identifier - Examples

Identifiers name	Valid	Invalid
8_okay	yes	
Hey.there.bye	no	
I<3	no	
intPay	yes	
#iLikePie	no	
$lollypop	yes	
^_^	no	
_8okay	yes	

Identifier Example - answers

Identifiers name	Valid	Invalid
8_okay		X
Hey.there.bye		X
I<3		X
intPay	X	
#iLikePie		X
$lollypop	X	
^_^		X
_8okay	X	

Naming convention

- We use naming convention to be consistency and for other people to understand that it is a variables , class
- We start with a lowercase letter for : variables, methods and objects. If it is a two word variable the second word become capitale
- Ex: applePie, spiderMan, bankReport
- For class, we start with an uppercase and use the same rules as above
- Ex: FirstPrograms, TestingProgram

Primitive data

- There are 8 primitive data type
- -Numeric :

  4 types for integers (ex: 8, 69)

  *byte, short, int, long

  2 types for floating-point(ex:2.3)

  *float, double
- -character (ex: A)
- *char
- -boolean (true or false)
- *boolean

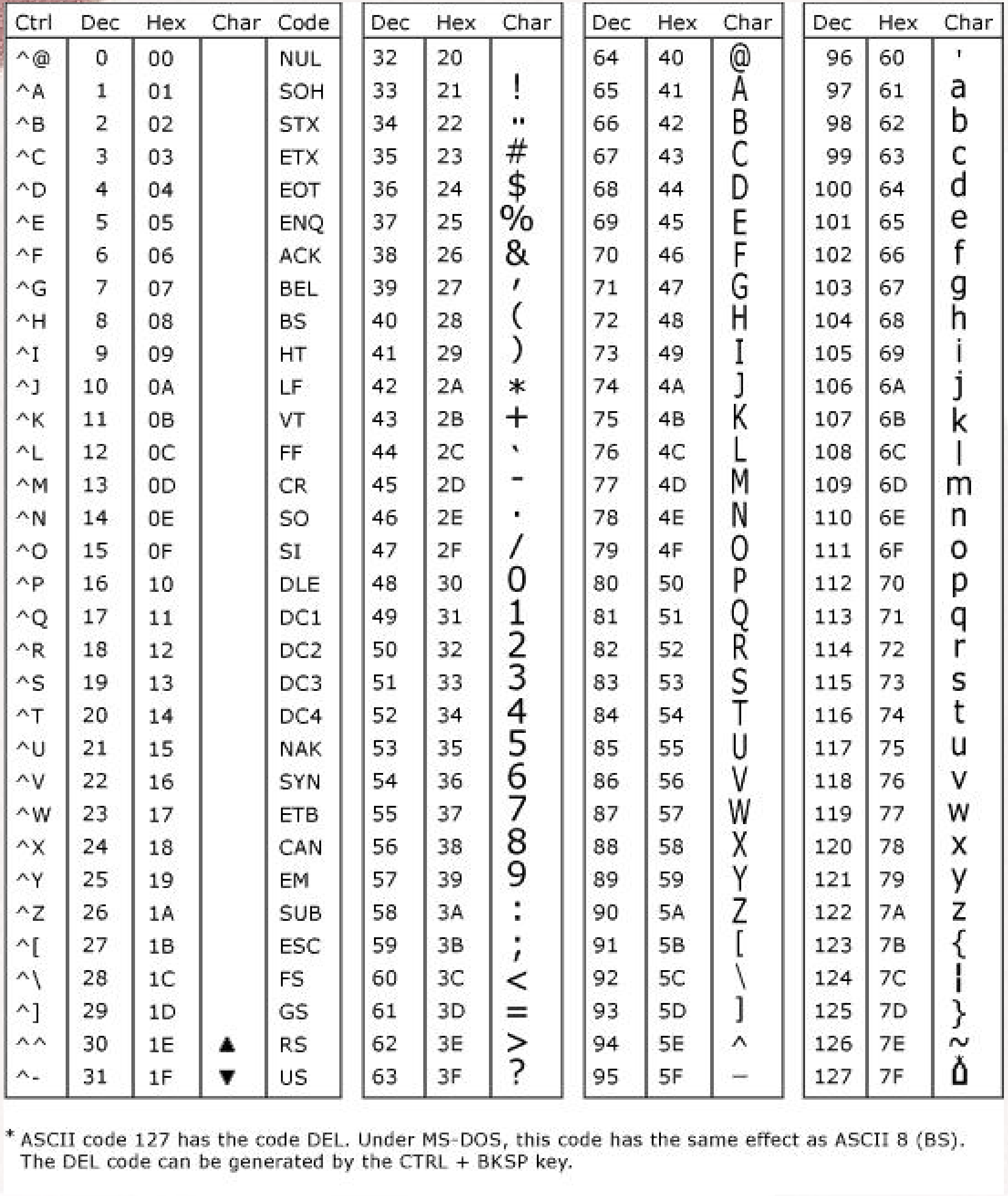
Floating point

- Floating point in mathematics and computer are different
- -in mathematically, 1/3 is equals to 0.3333333....
- -in computer, 1/3 is equal to 0.3333333333
- *it is due to limited space memory*

Characters

- We use the ascii code to represent character in computer
- We use the reserve word “char” to assign the type as character
- It can only hold one character (ex: “a”, “D”, “\n”)

Ascii table



Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20		64	40	@	96	60	'
^A	1	01		SOH	33	21	!	65	41	A	97	61	a
^B	2	02		STX	34	22	"	66	42	B	98	62	b
^C	3	03		ETX	35	23	#	67	43	C	99	63	c
^D	4	04		EOT	36	24	$	68	44	D	100	64	d
^E	5	05		ENQ	37	25	%	69	45	E	101	65	e
^F	6	06		ACK	38	26	&	70	46	F	102	66	f
^G	7	07		BEL	39	27	’	71	47	G	103	67	g
^H	8	08		BS	40	28	(	72	48	H	104	68	h
^I	9	09		HT	41	29	)	73	49	I	105	69	i
^J	10	0A		LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	-	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	.	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	0	80	50	P	112	70	p
^Q	17	11		DC1	49	31	1	81	51	Q	113	71	q
^R	18	12		DC2	50	32	2	82	52	R	114	72	r
^S	19	13		DC3	51	33	3	83	53	S	115	73	s
^T	20	14		DC4	52	34	4	84	54	T	116	74	t
^U	21	15		NAK	53	35	5	85	55	U	117	75	u
^V	22	16		SYN	54	36	6	86	56	V	118	76	v
^W	23	17		ETB	55	37	7	87	57	W	119	77	w
^X	24	18		CAN	56	38	8	88	58	X	120	78	x
^Y	25	19		EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
^[	27	1B		ESC	59	3B	;	91	5B	[	123	7B	{
^\	28	1C		FS	60	3C	<	92	5C	\	124	7C	¦
^]	29	1D		GS	61	3D	=	93	5D	]	125	7D	}
^^	30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
^-	31	1F	▼	US	63	3F	?	95	5F	_	127	7F	⌂*

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

Boolean

- Its value is true or false
- We use “boolean” as reserve word
- It is read like a book meaning top to bottom

Variables

- We use variables to store information (price, weight, size, etc...)
- It must be declared before we can use it
-
-

Examples

```
        int totalPrice;
    Int price1, price2;
```

Variable name

We can declare multiple variables in one line

Data type

Variables

- We can assign variables with value as we declare them
- *note: if it is declared but not assign a value, it will sometime set to default such as 0, Null or false, etc...

- Example

```
Int totalPrice = 100;
Int sum = 1 * 2 * 3;
Int number1 = 10, number 2 = 20;
```

Example of declaration & initialization

```
public class Cupcakes {
 public static void main(String[] args) {
  double price = 9.99;
  System.out.println("The price of cupcakes are $" + price + " each ");
  }
}
```

<u>Output</u>

```
The price of cupcakes are $9.99 each
```

Constant

- It can only hold one value
- Cannot change the value once is initialize
- User “final” modifier
- Example:
- final int TAX_RATE = 0.15;
- By convention, we put all capitals and seperate by _
- Pros:

  -Easy to update, prevent variables being changed

Cons:

  -Can’t change value of variable

print vs println

```
System.out.print("hello");
System.out.print("you");
System.out.println("hello");
System.out.println("you");
System.out.println();
int price = 50;
System.out.print(price);
char initial = 'L';
System.out.println(initial);
```

print vs println - answers

Output:

```
helloyouHello
you

50L
```

Escape sequences

Escape sequence	Meaning
\b	Backspace
\t	Tab
\n	New line
\”	Double quotation
\'	Single quotation
\\	Double backlash

What's the output of the following code?
System.out.print("one\ntwo\nthree\n");

How do you write the code to print below?
Read the file "c:\windows\readme.txt"

Escape sequence - asnwers

What's the output of the following code?
System.out.print("one\ntwo\nthree\n");
Answers: one two three

How do you write the code to print below?
Read the file "c:\windows\readme.txt"
System.out.println(“Read the file \”c:\\windows\\readme.txt\””);

Division and remainder

- Division for computer science is different than regular math, we only consider the whole number
- Example:

  1/3 = 0, while in math is 0.33333....

  10/8 = 1, in math is 1.25
- The remainder is the same concept
- Example:

  9 % 12 = 9

  25 % 5 = 0

Operator precedences

precedence	Examples
1st	Parenthese ( )
2nd	Unary: + and -
3rd	* , / , %
4th	Binary: + , -, concatenation
5th	Assignment operator =

Operator associative

- Unary: are grouped from right to left
- Example:
- +-+price = +-(+price) = +(-(+price))
- Binary: are grouped from left to right
- Example:
- price + rate + total = (price+rate) + total
- *only exception is assign operator
- Example:
- n1 = n2 = n3 ---> $\begin{tikzcd} n1 \arrow[r, phantom, "="] & n2 \arrow[l, bend left] \arrow[r, phantom, "="] & n3 \arrow[l, bend left] \end{tikzcd}$ --> n1 = n3

Shorthand Assignment Statements

- Shorthand assignments statements are used to simplify code and make it less redundant. This prevent errors

shortcut	Examples	equivalent
+=	X += Y	X = X + Y
-=	X -= Y	X = X - Y
*=	X *= Y	X = X * Y
/=	X /= Y	X = X / Y
%=	X %= Y	X = X % Y

Examples of shortcut assignment statements

```
int amount = 10;
Amount += 5;

System.out.println(amount);
double temp = 10;
temp *= 10;

System.out.println(temp);

String word = "hello"; word += "bye"; System.out.println(word); word
*= "bye";  // ???
```

Shortcut assignment statements - answers

`int amount = 10;`
`amount += 5;`
 	
`System.out.println(amount);`
`double temp = 10;`
`temp *= 10;`
 	
`System.out.println(temp);`
 	
`String word = "hello";`
`word += "bye";`
`System.out.println(word);`
	Output:
	15
	100.0
	hellobye

Increment and Decrement operator

- We can use the increment and decrements operator as shortcut to add/subtract
- Increment: (++) to add one from variables
- Decrements: (--) to subtracts one from variables
- Examples:
- count++ is same as count = count + 1
- count-- is same as count = count - 1

Increment and Decrement operator

- Prefix: ++price;
- -Will increment/decrements variable by 1
- -add plus 1 to price then give output

- Postfix: price++;
- -Will increment/decrements variable by 1
- -give output of price then add plus 1

Increment/decrement examples

```
              int prefix = 20;
              int postfix = 10;


       System.out.println("Before");

       System.out.println(++prefix);

       System.out.println(postfix--);


       System.out.println("After");

       System.out.println(prefix);

       System.out.println(postfix);
```

Automatically change data type

- In java, operand will convert all data type into one type if they all have different types.
- Rules
- -double ------> double
- -float ----------> float(double)
- -long ----------> long
- -short, byte, char ---------> int
- Example:
- aInt + aByte = int
- aLong – aDouble * aInt = double

Auto change data type - examples

```
      short aShort = 2;
            int aInt = 23;
            long aLong = 10;

            float aFloat = 1.0f;
            double aDouble = 2.5;

            System.out.println(aShort +
aInt + aLong);
            System.out.println(aShort +
aLong + aFloat);
            System.out.println(aDouble *
aLong + aShort);
```

Output:

```
35
13.0
27.0
```

Casting

- We can force a conversion by casting
- **Syntax**: (desired data type) variable_name
- Example:
- double num1;
- (int) num1 = 3.8;
- System.out.println(num1);

Output

3

String

- A string is a reference to an object, basically a string is a collection of character that can form a “object” or known as word/sentence

Example

```
        System.out.println(“hello”);
        System.out.println('A');
```

Declaring String

create a reference for name

```
String name;
```

Constructor of type String

```
name = new String("name of the person");
```

(=) Connect reference to object

(new) Create an object of type String associated with name

Declaring String

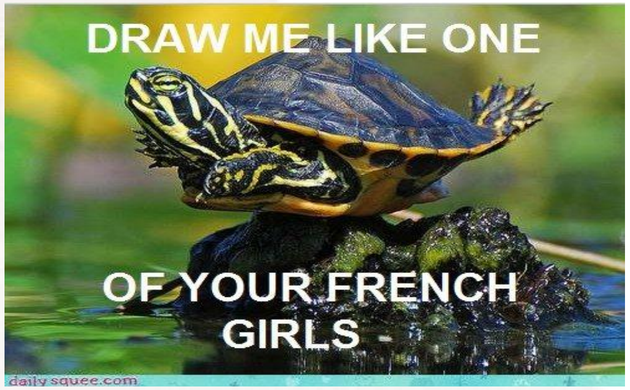
- Since String is commonly used, we do not need to connect the reference and object, instead we can simplify it
- Example:
-

With new operator	Without new operator

String name;
name = new String("Obama");
	String name;
	name = "Obama"

String concatenation

- String are immutable meaning that it cannot be changed once is created (cannot shorten/lengthen and modify contents)
- We can combine multiple string, using the “+” operator
- Example:
- String greeting = “Hello, ” + “I like” + “ turtles”;
- System.out.println(greeting);
- Output:
- Hello, I like turtles



String methods

- We can use many methods with string such as...
- -**length()** : return the length of string
- -**toUpperCase()**: turn string into all uppercase
- -**toLowerCase()**: turn string into all lowercase
- -(char) **charAt(#)**: return the character of string given the position

  -(int) **indexOf(String)**: return the index of string given a string, return -1 if not found

  -(boolean) **equals(other_string)**: return true if both string contains the same contents, else false
- *more on page 38

Return
ype

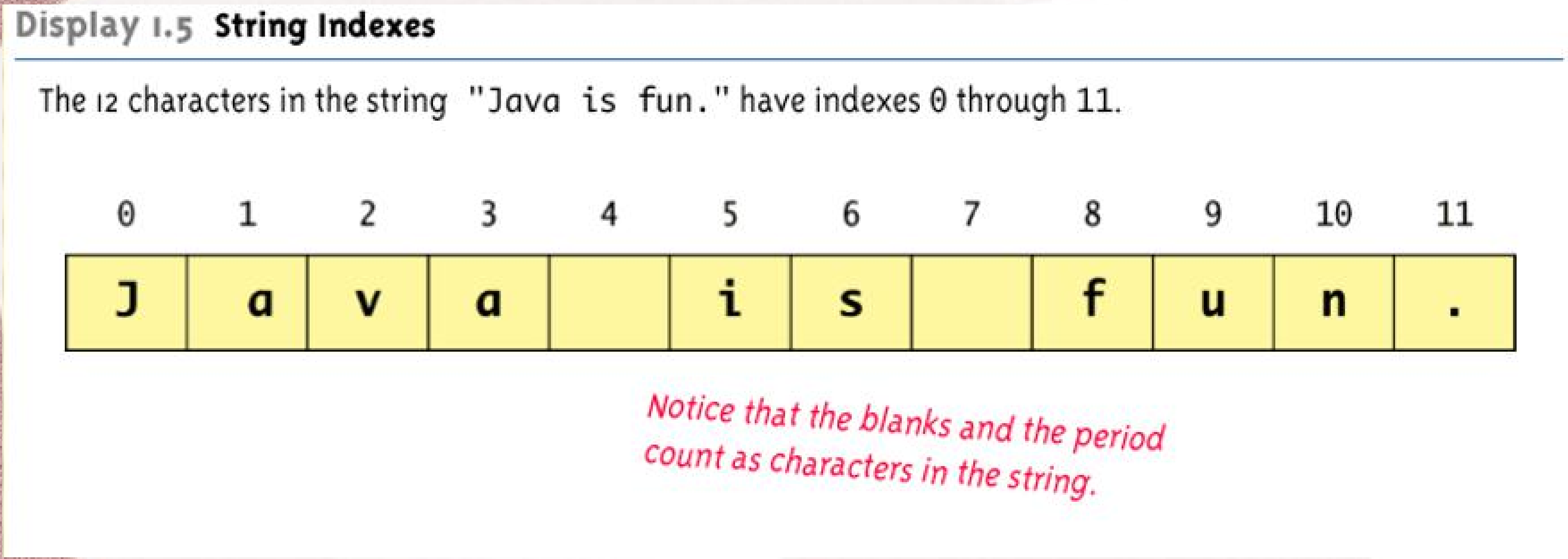
How to use string methods

```
String s1 = "This string length is 24";
String s2 = "I like creampies";
String s3 = "I like to eat big banana";

System.out.println("==Finding the length==");
System.out.println(s1.length());

System.out.println("==charAt example==");
System.out.println(s2.charAt(2));

System.out.println("==IndexOf example==");
System.out.println(s3.indexOf("like"));
```

Output:

```
==Finding the length==
24
==charAt example==
l
==IndexOf example==
2
```

String start at index zero


Display 1.5 **String Indexes**

The 12 characters in the string "Java is fun." have indexes 0 through 11.

0	1	2	3	4	5	6	7	8	9	10	11
J	a	v	a		i	s		f	u	n	.

Notice that the blanks and the period count as characters in the string.