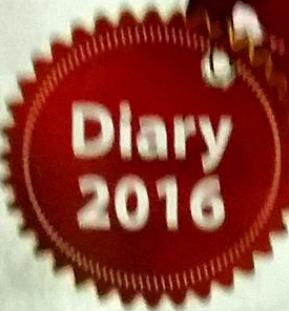


Java BASIC



Diary
2016

For :

Leads Java SE_Android Batch 2



Graphic Arts Institute

January 2016

Graphic Arts Institute



05

Tuesday

Time

Januray 2016

Core Java → Language Fundamentals

- ① Identifiers
- ② Reserved Words
- ③ Data Types
- ④ Literals
- ⑤ Arrays
- ⑥ Types of variables
- ⑦ var-arg methods

06

Wednesday

Time

Januray 2016

- ⑧ main method
- ⑨ Command Line arguments
- ⑩ Java coding standards

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31		1			
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

January 2016

Graphic Arts Institute



Januray 2016

07

Thursday

Time

Identifiers:

A name in java program is called identifier. Which can be used for identification purpose. It can be method name, class name, variable name or table name.

Example:

08

Friday

Time

Januray 2016

class Test

public static void main(String[] args)

{

~~System.out.println("Hello")~~

int x = 1;

}

5

① ② ③ ④

January 2016						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

How many identifiers are here: 5

January 2016

Graphic Arts Institute



09 Saturday

Time

Januray 2016

① Rules for defining Java Identifiers:

① The only allowed characters in Java identifiers are:

a to z
A to Z
0 to 9
\$
_

total_number ✓ (Valid)
total# ✗ (Invalid)
total123 ✓ (Valid)
123total ✗ (Invalid)

If we will use any other character

10 Sunday

Time

Januray 2016

Then we will get compile time error.

② Identifiers can't start with digit.

total123 ✓

123total ✗

③ Java Identifiers are case sensitive

int valid = 1;
int Valid = 1;
int VALID = 1; } { all are
different
variable.

January 2016
Sat Sun Mon Tue Wed Thu Fri
30 31
1 2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29



January 2016

Januray 2016

11 Monday

Time

4) There is no length limit for java identifiers. But it is not recommended to take too lengthy identifiers.

⑤ We can't use reserved words as identifiers.

int $x = 10;$ ✓

int if = 10; ✗

↙ this is reserved word

12 Tuesday

Time

Januray 2016

⑥ All predefined java class names and Interface names, we can use as identifiers

class Test {

 public static void main (String[] args) {

 int String = 88; int Runnable = 99;

 S.O. pln(String); S.O. pln(Runnable);

January 2016

Sat Sun Mon Tue Wed Thu Fri

30 31 1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

6 17 18 19 20 21 22

3 24 25

Even though it is valid but it is not a good programming practice, bcz it reduces readability and creates confusion.



13

Wednesday

Time

Januray 2016

Reserved Words:

In java some words are reserved words, to represent some meaning and functionality. Such type of words are called, reserved words.

Reserved Words (53)

Key Words (50)

Reserved Literals (3)

14

Thursday

Time

Januray 2016

Used Keyword (48)

Unused keyword (2)

if else ---

goto const

→ true
→ false
→ null

January 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31		1			
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

January 2016

Graphic Arts Institute



15 Friday

Time

Januray 2016

Keywords for Data types:(8)

byte

short

int

long

float

double

boolean

char

16

Saturday

Time

Januray 2016

Keywords for flow control:

if

else

switch

case

default

while

do

for

break

continue

return

January 2016

Sat Sun Mon Tue Wed Thu Fri

30 31

1 2 3 4 5 6 7

8 9 10 11 12 13 14 15

16 17 18 19 20 21

22 23 24 25 26 27 28 29

30

January 2016

Graphic Arts Institute



17 Sunday

Time

Januray 2016

Keywords for modifiers: (11) with default
it is (12)

public

private

protected

static

final

abstract

synchronized

native

strictfp (1.2 v)

18 Monday

Time

Januray 2016

transient

volatile

Keywords for exception Handling (6)

try

catch

finally

throw

throws

assert (1.4 v)

January 2016

Sat Sun Mon Tue Wed Thu

30 31

1

2 3 4 5

6 7 8 9 10 11 12

13 14 15

16 17 18 19 20 21

22 23 24 25 26 27 28

29

January 2016

Graphic Arts Institute



19

Tuesday

Time

Januray 2016

Class Related Keywords: ⑥

class

Interface

Extends

Implements

Package

Import

Object

return

String

String

int

boolean

boolean

String

(void) method

Januray 2016

20

Wednesday

Time

Object Related keywords: ⑦

new

instanceof

super

this

Return type: ①

January 2016

Sat Sun Mon Tue Wed Thu Fri

30 31

1

2 3 4 5

8

9 10 11 12 13 14 15

15

16 17 18 19

20 21 22

23 24 25 26 27 28 29

void

In Java return type is mandatory.
If a method won't return anything
then we have to declare that method
void - return type.

January 2016

Graphic Arts Institute



21

Thursday

Time

Januray 2016

Unused Keywords: ②

goto: Uses of goto, create a several problems, in old languages and hence some people ban this keyword in java.

const: Use of final instead of "const".

N.B! goto and const are unused

22

Friday

Time

Januray 2016

Keywords and if we are trying to use, we will get compile time error.

Reserved Literals: ③ true, false, null

true and false are values for boolean data types.

January 2016

Sat Sun Mon Tue Wed Thu Fri

30 31

1

2 3 4 5 6 7 8

9 10 11 12 13 14

15 16 17 18 19 20 21 22

23 24 25 26 27 28 29

null is default value for object reference,

January 2016

Graphic Arts Institute



23

Saturday

Time

Januray 2016

Enum keyword : ①

enum (1.5 v)

We can use enum to define
a group named constant.

Example :

enum month

{

jan, feb, mar, ... Dec ;

24

Sunday

Time

Januray 2016

enum deer {

KP, KO, RC ... ,

}

January 2016

Sun Mon Tue Wed Thu Fri

31 1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

January 2016

Graphic Arts Institute



25 Monday

Time

Januray 2016

Conclusions:

- ① All 53 reserved words in java contains only lower case alphabet & simbles.
- ② In java we have only new keyword, and there is no ~~no~~ delete keywords , bcz destruction of user's object is the responsibility of garbage collector .

26 Tuesday

Time

Januray 2016

- ③ The following are new keywords in java :

strictfp 1.2 ✓

assert 1.4 ✓

enum 1.5 ✓

January 2016

Sun Mon Tue Wed Thu Fri

30 31

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

January 2016

Graphic Arts Institute

27

Wednesday

Time

Januray 2016

- ① strictfp but not strictfp
instanceof but not instanceof
synchronized but not synchronize
extends but not extend
implements but not implement
import but not imports
const but not constant

28

Thursday

Time

Januray 2016

January 2016						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
31		1				
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

January 2016

Graphic Arts Institute

29

Friday

Time

Januray 2016

Data types:

1. In java every variable and every expression has some type. Each and every data type is clearly defined.

Every assignment should be checked by compiler for type compatibility.

Bcz of above reasons we can conclude java language is strongly typed programming language.

30

Saturday

Time

Januray 2016

2. Java is not considered as pure object oriented programming language bcz several OPP features are not satisfied by java. (Like: operator overloading, multiple Inheritance etc.)

Moreover we are depending on primitive data types, which are non objects.

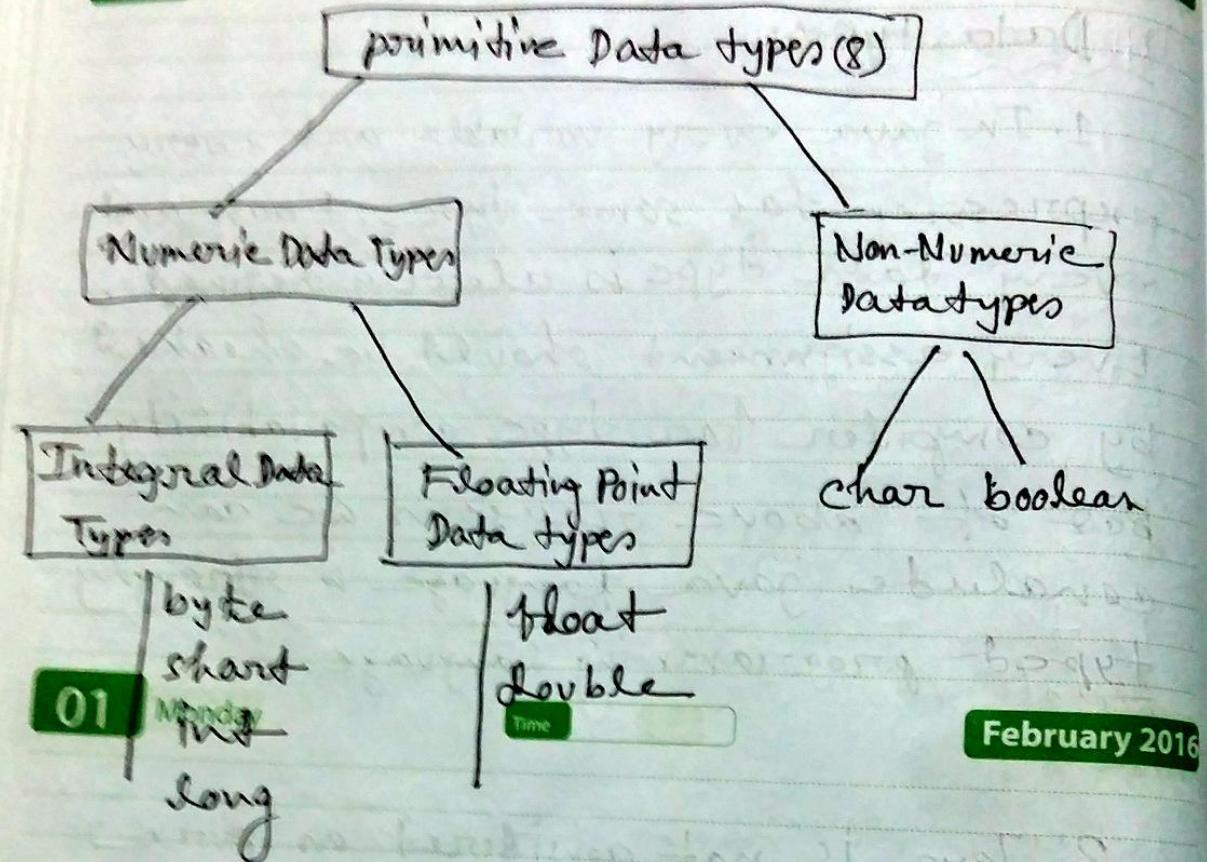
January 2016
Sat Sun Mon Tue Wed
30 31
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15
16 17 18 19 20 21
22 23 24 25 26 27 28 29



31 Sunday

Time

Januray 2016



Except boolean and char, remaining data types ~~are~~ are considered as signed data types, bcs we can represent both positive and negative numbers.

January 2016
Mon 01 Tue 02 Wed 03 Thu 04 Fri 05 Sat 06 Sun 07
1 2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

```

int a = -10;
int b = +5;
float c = -10.5;
char d = -'a'; X
boolean b = -true; X
  
```



02 Tuesday

Time

February 2016

① byte: Size 1 byte (8 bit)

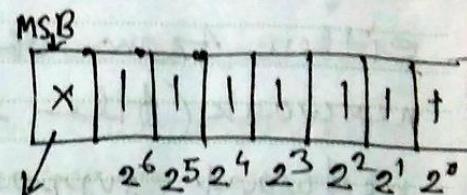
Max. Value $\rightarrow +127$

Min. Value $\rightarrow -128$

Range: $-128 \text{ to } +127$

Negative value

calculate always with
seconds complement.



signbit $64+32+16+8+4+2+1$

0 $\rightarrow +ve = 127$

1 $\rightarrow -ve$

03 Wednesday

Time

February 2016

✓ byte b = 10;

✓ byte b = 127;

✗ byte b = 128;

{CE: Possible loss of precision

found: int

required: byte

byte b = true;

{CE: incompatible types

found: boolean

required: byte

✗ byte b = 10.5;

{CE: Possible loss of precision

found: double

required: byte

byte b = "stab";

{CE: incompatible types

found: String

required: byte

04

Thursday

Time

February 2016

Byte is the best choice, if we want to handle data in terms of streams either from the file or from the network (file supported form on network supported form is byte).

II Short:

Size : 2 bytes (16 bits)

Range : -2^{15} to $2^{15} - 1$

05

Friday

Time

February 2016

$[-32768 \text{ to } 32767]$

This is the most rarely used data types in java.

v short s = 32767;

x short s = 32768;

CE : PLP

found: int

required: short

short s = true; 2 : X

CE : incompatible type

found: boolean

required: short

06 Saturday

Tues

February 2016



short data types is best suitable for 16 bit processor like 8085 but these processors are completely outdated and hence corresponding short data type is also ~~is~~ outdated data types.

07 Sunday

Tues

February 2016

III int:

size \rightarrow 4 byte (32 bit)

Range \rightarrow -2^{31} to $2^{31}-1$

$[-2147483648 \text{ to } 2147483647]$

The most commonly used data types in java is "int".

February 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29

February 2016

February 2016

08 Monday

✓ int $x = 2147\text{,}3648^{48}$;

✗ int $x = 2147\text{,}483648^{48}$;
CE: integer number too large

✗ int $x = 2147\text{,}483648482$;

CE: PLP
found: long
required: int

09 Tuesday

February 2016

✗ int $x = \text{true};$;

CE: incompatible type

found: boolean

required: int

Sometimes

February 2016

Sun Mon Tue Wed Thu Fri Sat

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29



10 Wednesday

Time

February 2016

⑩ Long: sometimes int may not enough to hold big values then we should go for long type:

Ex: The amount of distance traveled by lights in 1000 days, to hold this value int may not enough, we should go for long data type.

$$\text{long } l = 126,000 \times 60 \times 60 \times 24 \times 1000 \text{ miles}$$

11 Thursday

Time

February 2016

Ex2: The number of characters present in a big file may exceeds int range, hence the return type of length method is long but not int.

$$\text{long } l = f.length();$$

February 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29

size: 8.bytes (64 bit)

Range: -2^{63} to $2^{63}-1$

12

Friday

Time

February 2016

All the above data types (byte, short, int, long) represent integral values. If we want to represent floating point values, then we should go for floating point data types.

13

Saturday

Time

February 2016

Floating Point data types:

Float

① 5 to 6 decimal points

⑪ Single precision
(less accuracy)

⑬ 4 bytes

February 2016

⑭ Range!

Sat	Sun	Mon	Tue	Wed	Thu	Fri
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

-3.4e38 to 3.4e38

Double

① 15 to 20 after decimal point

⑫ Double precision
(more accuracy)

⑬ 8 bytes

⑭

-3.4e308 to 3.4e308

-1.7e308 to 1.7e308

February 2016

Graphic Arts Institute



14

Sunday

Time

February 2016

✗ boolean:

Size: N/A (Not applicable) VM dependent

Range: N/A (Not applicable)

but allowed values are: true/false

✓ boolean b = true;

✗ boolean b = 0; ↴

CE: incompatible types

found: int

required: boolean

15

Monday

Time

February 2016

✗ boolean b = True; ↴

CE: can not find symbol

symbol: variable True

location: class Test

✗ boolean b = "True"; ↴

CE: incompatible type

found: java.lang.String

required: boolean

February 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29

February 2016

Graphic Arts Institute



February 2016

16 Tuesday

Time

1. int $x = 0;$

```
if ( $x$ ) {  
    S. O. P ("Hello");  
} else {  
    S. O. P ("Hi");  
}
```

CE: incompatible types

found: int
required: boolean

2. while (1) {

```
    S. O. P ("Hello");  
}
```

17

Wednesday

Time

February 2016

Both example are valid in C ~~but not~~ (most languages), but not in java bcz java is strongly typed language.

February 2016

Sun Mon Tue Wed Thu Fri

1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

February 2016

Graphic Arts Institute



18

Thursday

Time

February 2016

18 char:

Old Languages (Like c, c++) are ASCII code based and the number of allowed different ASCII code characters are ≤ 256 . To represent these 256 characters 8 bits are enough. Hence the size of char in old language is 1 byte(8 bit)

19

Friday

Time

February 2016

But java is UNI code based and the number of different unicode characters are > 256 and ≤ 65536 . To represent these many characters 8 bits may not enough. Compulsory we should go for 16 bit(2 byte). Hence the size of char in java is 2 bytes.

February 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29

Range: 0-65535

February 2016

Graphic Arts Institute



20 Saturday

Time

February 2016

Summary of Java Primitive Data Types

Data Type	Size	Range	Wrapper Class
byte	1 byte	-2^7 to $2^7 - 1$	Byte
short	2 bytes	-2^{15} to $2^{15} - 1$	Short
int	4 bytes	-2^{31} to $2^{31} - 1$	Integer
long	8 bytes	-2^{63} to $2^{63} - 1$	Long
float	4 bytes	-3.4×10^{-38} to 3.4×10^{-38}	Float

21 Sunday

Time

February 2016

double	8 bytes	-1.7×10^{-38} to 1.7×10^{-38}	Double
boolean	N/A	N/A (but allows 2 values are true, false)	Boolean
char	2 bytes	0 to 65535	character

February 2016
 1 2 3 4 5
 6 7 8 9 10 11 12
 13 14 15 16 17 18 19
 20 21 22 23 24 25 26
 27

February 2016

Graphic Arts Institute

22

Monday

Time

February 2016

Default value

0

0

0

0

0.0

23

Tuesday

Time

February 2016

false

0 [represents
space
character]

February 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29

February 2016

24

Wednesday

Time

February 2016

~~null~~

char ch = null;



CE; incompatible types

found: null type

required: char

null

~~^~~ is the default value for
object reference. And we can't

25

Thursday

Time

February 2016

apply for primitives. If we are
trying to use for primitive, then
we will get compile time error.

February 2016						
Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29				

February 2016

Graphic Arts Institute

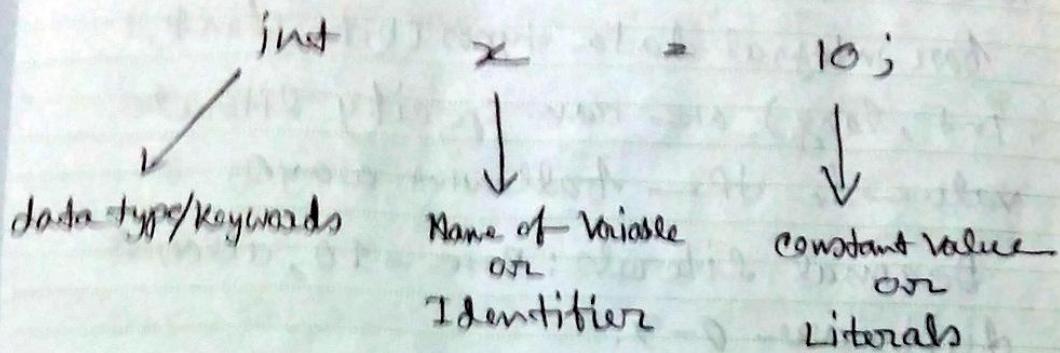
26 Friday

Time

February 2016



17th ~~Variables~~ Literals:



A constant value which can be assigned to the variable is called a literal.

27 Saturday

Time

February 2016

Integral Data type Literals value:

byte
short
int
long

① decimal literals (Base-10)

int x = 10; 0-9

② octal literals (Base-8)

int x = 010; 0-7

↓
prefix will be 0 for octal

③ Hexadecimal literals (Base - 16)

int x = 0x10; 0-9
a-f

↓
prefix will be 0x for hexadecimal

February 2016

1	3	5	7	9	11	13
6	8	10	12	14	16	18
15	17	19	21	23	25	27
28	29					

February 2016

February 2016

28 Sunday

Time

For Integral literals:

For integral data types (byte, short, int, long) we can specify literal value in the following ways.

Decimal literals: Base - 10, allowed digits are 0-9.

For extra digit in Hexadecimal

(a to f) we can use both lower or upper case letters. These

29 Monday

Time

February 2016

In one of very few areas where Java is not case sensitive the literal value should be prefixed with 0x or Ox.

✓ int x = 10;

✗ int x = 0786; CE: integer number too

large
In octal 8 is not allowed.

✓ int x = 0xFace;

✗ int x = 0777;

Feb Sun Mon Tue Wed Thu

1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29

Carried over from X0 2015 Notes

March 2016

Graphic Arts Institute

01 Tuesday

March 2016

class Test {

public static void main(String[] args) {

int x = 10;

$$(10)_8 = (?)_{10}$$

int y = 010;

$$\begin{aligned}0 \times 8^0 + 1 \times 8^1 \\= 0 + 8 = 8\end{aligned}$$

int z = 0x10;

System.out.println(x + "..." + y + "..." + z); (10)_16 = (?)_10

}

$$\begin{aligned}0 \times 16^0 + 1 \times 16^1 \\= 0 + 16 = 16\end{aligned}$$

}

output: 10....8...16

02

Wednesday

March 2016

10
010
0x10

→ int

10L
010L
0x10L

→ long

By default every integral literals are int type. But we can specify explicitly or long type by suffix with small "L" on

1	2	3	4
5	6	7	8
9	10	11	
12	13	14	15
16	17	18	
19	20	21	22
23	24	25	
26	27	28	29
30	31		

March 2016

March 2016

03 Thursday

Time

 $\checkmark \text{int } x = 10;$ $\checkmark \text{long } l = 10L;$ $\times \text{int } x = 10L; \rightarrow$ $\checkmark \text{long } l = 10;$

CE: PLP
 found: long
 required: int

04 Friday

Time

March 2016

There is no direct way to specify byte and short explicitly. But indirectly we can specify. Whenever we are assigning integral literals to the byte variable and if the value is in the range of

~~byte~~, then compiler treats

it automatically as byte literals.
similarly, short literals also.

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4

5 6 7 8 9

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

March 2016

Graphic Arts Institute



05

Saturday

Time

March 2016

✓ byte b = 10;

✓ byte b = 127;

✗ byte b = 128; →

{CE: PLP}

found: int

required: byte

✓ short s = 32767;

✗ short s = 32768; →

{CE: PLP}

found: int

required: short

06

Sunday

Time

March 2016

March 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

March 2016

Graphic Arts Institute

07 Monday

Time

March 2016

Floating Point literals: By Default

every floating point literals are double type and hence we can't assign to the float variable. But we can specify floating point literal as float type by suffix with "f" or "F".

08 Tuesday

Time

March 2016

X float f = 123.456; CE: PLP

✓ float f = 123.45f;

Found: double
Required: float

✓ double d = 123.56;

We can specify explicitly floating point literals as double type by suffix with "d" or "D". Of course this convention is not required.

March 2016

09 Wednesday

Graphic Arts Institute



March 2016

✓ double d = 123.456 D;

✗ float f = 123.453;

{CE: PL P
found: double
required: float}

We can specify floating point literals only in decimal form. and we can't specify in octal or hexadecimal

10 Thursday

Time

March 2016

forms:

✓ double d = 123.456 ;

✓ double d = 0123.456 ; {This will treated as decimal}

✗ double d = 0x123.456 ;

||

CE: malformed decimal point literals.

March 2016

1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

March 2016

Graphic Arts Institute

11

Friday

Time

March 2016

We can assign integral literal directly to floating point variables and that integral literals can be specified, either in decimal or octal or hexadecimal forms.

✗ double d = 0786; \rightarrow CE: integer number too large

✓ Double d = 0xFace;

✓ Double d = 0786.0;

12

Saturday

March 2016

✗ double d = 0xFace.0; double d = 10;

✓ Double d = 10;

s. o. pln(d);

✓ Double d = 0277;

output: 10.0

[best, integral decimal number converted into floating point number;]

March 2016

Sun Mon Tue Wed Thu Fri

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

March 2016

Graphic Arts Institute

13 Sunday

Time

March 2016

We can't assign floating point literal to integral types;

✓ double d = 10;

✗ int x = 10.0; } CE: PLP

found: double
required: int

14 Monday

Time

March 2016

We can specify floating point literals, even in exponential form ($e: 1.2e3$)

✓ double d = 12e3; $\rightarrow 12e3 = 12 \times 10^3 = 1200.0$
gets a double number

✗ float f = 1.2e3; } we can not assign float to double variable
CE: PLP
found: double
required: float

March 2016

Sun Mon Tue Wed Thu Fri

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

March 2016

March 2016

15

Tuesday

Time

~~15~~ boolean literals: the only allowed values for boolean data types ~~are~~ true/false.

✓ boolean b = true;
 ✗ boolean b = 0; → {OE: incompatible
found: int
required: boolean}

16

Wednesday

Time

March 2016

✗ boolean b = True; → {OE: can not find symbol
symbol: variable True
location: class test}

✗ boolean b = "true"; → {OE: incompatible
types
found: String
required: boolean}

March 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
		1	2	3	4	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

March 2016

17 Thursday

Graphic Arts Institute



March 2016

```
int x = 0;  
if (x){  
    S. O. pln("Hello");  
}else{  
    S. O. pln("Hi");  
}
```

CE: incompatible type
found: int
required: boolean

18 Friday

March 2016

```
while (1){  
    S. O. pln("Hello");  
}
```

But both are valid in C
language.

March 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4			
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

March 2016

Graphic Arts Institute



19 Saturday

Time

March 2016

Character literal:

We can specify char literal as single character within single quotes.

✓ `char ch = 'a';`

✗ `char ch = a;`

CE: can not find symbol
symbol: variable a
location: class test 2

20 Sunday

Time

March 2016

✗ `char ch = "a";`

CE: incompatible type
found: J. S. String
required: char

✗ `char ch = 'ab';`

CE1: unclosed char literal
CE2: unclosed char literal
CE3: not a statement

March 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4		
5	6	7	8	9	10
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31			

March 2016

Graphic Arts Institute



21

Monday

Time

March 2016

We can specify char literal as integral literal which represents unicode value of the character and the integral literal can be specified in either decimal, octal or hexadecimal form. But allowed range is 0 to 65535

22

Tuesday

Time

March 2016

✓ `char ch = 97;`

`s.o.p(ch);` output: a

✓ `char ch = 0xface;`

✓ `char ch = 0777;`

✓ `char ch = 65535;`

✗ `char ch = 65536;`

March 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4			
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

CB: PLP
found: int
required: char

March 2016

Graphic Arts Institute

23

Wednesday

March 2016

We can represent char literal
in unicode representation, which is
nothing but ~~single character~~

'\uXXXX'

↓
4-digit

hexadecimal
Number

char ch = '\u0061';

S. O. Put(ch);

Output: a

$$16 \begin{array}{r} 97 \\ -1 \end{array}$$

24

Thursday

March 2016

Every escape character is a valid
char literal.

✓ char ch = '\n';

✓ char ch = '\t';

✗ char ch = '\m'; } CE: illegal escape
character

March 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4			
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

March 2016

Graphic Arts Institute

25 Friday

Time

March 2016

In Java there are eight escape character:

escape character

Description

\n → new line

\t → Horizontal tab

\r → carriage return

\b → Back space

\f → Form feed

26 Saturday

Time

March 2016

' → Single quote

" → Double quote

\ \ → Back slash

March 2016

Su Sun Mon Tue Wed Thu Fri

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

March 2016

Graphic Arts Institute



27 Sunday

Time

March 2016

Which of the following are valid:

- char ch = 65536;
- char ch = ox Beer;
- char ch = 'Hello';

28 Monday

Time

March 2016

March 2016

Sun Mon Tue Wed Thu Fri
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

March 2016

29 Tuesday

Graphic Arts Institute



March 2016

String Literal:

Any sequence of characters within double quote is treated as string literal.

Ex:

String s = "durga";

30

Wednesday

March 2016

1.7 v Enhancement with respect two literals:

(1) Binary literals.

Four integral data types (byte, short, int, long)
until 1.6 v,
we can specify literal value in the
following ways:

March 2016

① decimal ② octal ③ hexadecimal
but 1.7 v onwards we can specify
literal value even in binary form
also.

March 2016

Graphic Arts Institute

31

Thursday

Time

March 2016

Allowed digits are : 0, 1

literal value should be prefixed with OB or OB.

Ex: int x=OB 1111;

S. O. P (x);

output: 15

01

Friday

Time

April 2016

① Uses of underscore symbol in numeric literals:

From 1.7 onwards we can use underscore symbol between digits of numeric literals.

March 2016

Sun Mon Tue Wed Thu Fri
1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

Ex: double d = 123_456_789;

or

double d = 1_23_456_7_8_9;

double d = 123_456_7_8_9;

April 2016

02 Saturday

Time

Graphic Arts Institute



April 2016

The main advantage of this approach is readability of the code will be improve.

At the time of compilation these underscore sign will be remove automatically. Hence after compilation the above line will become:

double d = 123456,786;

03 Sunday

Time

April 2016

We can use more than one underscore symbol between the digits.

✓ { double d = 1_ - 23 _ _ 4_ 5_ 6_ , 7_ 8_ 6; }
double d = 1_ - 23 _ - 4_ 5_ 6_ , 7_ 8_ 6;

April 2016

Mon Tue Wed Thu Fri

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	

April 2016

Graphic Arts Institute

04 Monday

Time

April 2016

We can use underscore symbol only between the digits. If we are using any other area, we will get compilation error.

X

} double d = 1_23_456,7_8_9;
double d = 1-23-456) ,78-9;
double d = 1-23-456,7-8-9();

05

Tuesday

Time

April 2016

April 2016

Mon Tue Wed Thu Fri

1

2 3 4 5 6

7 8 9 10 11

12 13 14 15

16 17 18 19 20

21 22 23 24 25

April 2016

Graphic Arts Institute

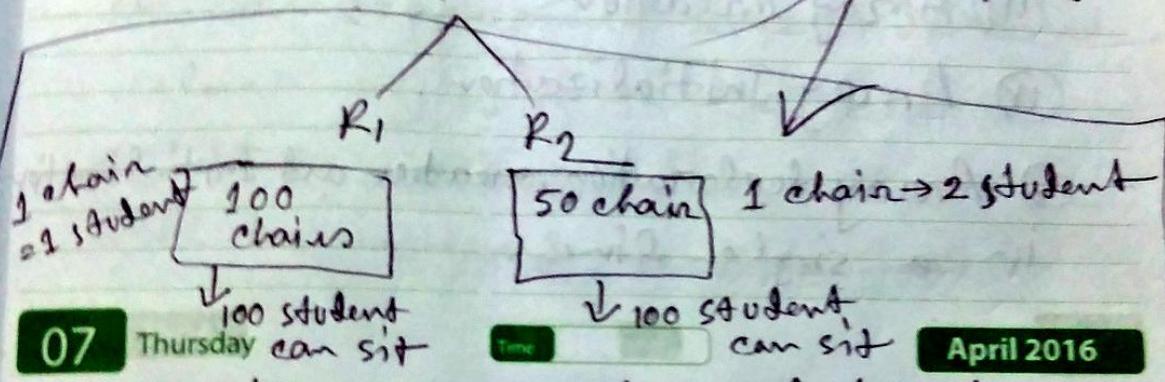
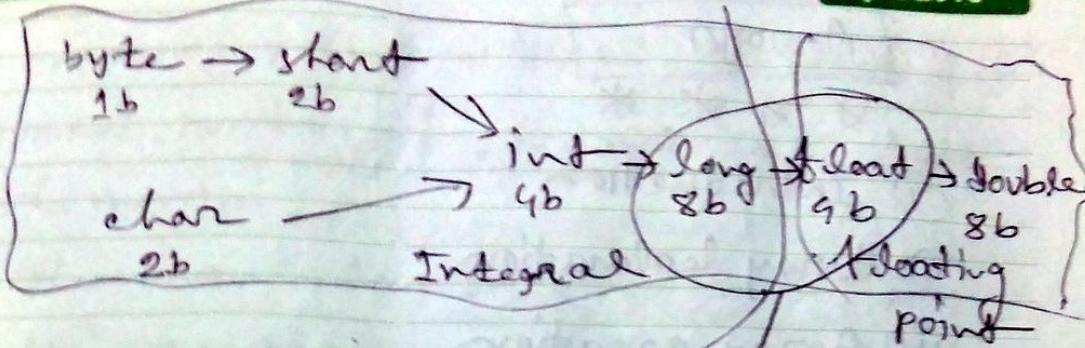


06

Wednesday

Time

April 2016



07

Thursday

Time

April 2016

This is same as long and float

8 byte long value we can assign
 4 byte float variable bcz both
 are following different memory
 representation internally.

float f = 10.2;

s. o. p (f);

output: 10.09

April 2016

Su	Mo	Tu	We	Th	Fr
1					
2	3	4	5	6	7
8	9	10	11	12	13
15	16	17	18	19	20
22	23	24	25	26	27
29					

April 2016

Graphic Arts Institute

08

Friday

Time

April 2016

Arrays:



- ① Introduction
- ② Array Declaration
- ③ Array creation
- ④ Array Initialization
- ⑤ Array declaration, creation and Initialization
in a single line.

09

Saturday

Time

April 2016

- ⑥ length vs length()
- ⑦ Anonymous Arrays
- ⑧ Array element assignments
- ⑨ Array variable assignment.

April 2016

Sun	Mon	Tue	Wed	Thu	Fri
					1
3	4	5	6	7	8
10	11	12	13	14	15
17	18	19	20	21	22
24	25	26	27	28	29

April 2016



Graphic Arts Institute



10 Sunday

Time

April 2016

An array is an indexed collection of fixed number of, Homogeneous data elements.

The main advantage of array is, we can represent huge number of values by using single variable, so that readability of the code will be improved, but the main

11 Monday

Time

April 2016

This advantage of array is fixed in size, that is once we create an array there is no chance of increasing or decreasing the size, based of our requirements. Hence to use arrays concept,

compulsory we should know the size in advance, which may not possible always.

April 2016

Sun Mon Tue Wed

3 4 5 6 7

10 11 12 13 14 15

17 18 19 20 21 22

24 25 26 27 28 29

April 2016

Graphic Arts Institute

12

Tuesday

Time

April 2016

Array declaration!

One-dimensional array declaration

int[] x;

int []x;

int x[];

13

Wednesday

Time

April 2016

int[] x; is recommended bcz
name is clearly separated from type.

At the time of declaration we can't
specify the size, otherwise we will
get compile time error.

int[6] x; X

int[] x; ✓

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

2016

April 2016

14 Thursday

Graphic Arts Institute

April 2016

Two Dimensional Array Declaration:

```

int [][] x;
int      x[][];
int      x[][];
int []   x[];
int []   x[][];
int      x x[][];
  
```



15 Friday

April 2016

- ✓ int [] a, b; $a \rightarrow 1$
 $b \rightarrow 1$
- ✓ int [] a[], b; $a \rightarrow 2$
 $b \rightarrow 1$
- ✓ int [] a[], b[]; $a \rightarrow 2$
 $b \rightarrow 2$
- ✓ int [] []a, b; $a \rightarrow 2$
 $b \rightarrow 2$
- ✓ int [] []a, b[]; $a \rightarrow 2$
 $b \rightarrow 3$

April 2016

Sat Sun Mon Tue Wed Thu Fri

30 1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25

~~int [] []a, []b; $\rightarrow CE$.~~

If we want to specify dimension before the variable, that facility is applicable only for first variable in declaration.

April 2016

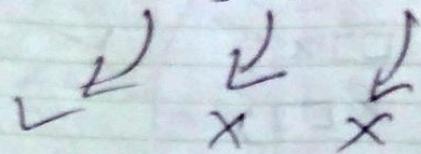
Graphic Arts Institute

16 Saturday

April 2016

If we are trying to apply for remaining variable, we will get compile time error.

```
int[] []a, []b, []c;
```



17 Sunday

April 2016

Three dimensional Array Declaration:

```
int[][][] a;
```

```
int
```

```
int[]
```

```
int[] a;
```

```
int[] a[];
```

```
int[] a[];
```

April 2016

Sur Sur Ben The End The Pg

30 2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

20 21 22 23 24 25 26

27 28 29 30 31 32 33

34 35 36 37 38 39 40

41 42 43 44 45 46 47

48 49 50 51 52 53 54

56 57 58 59 60 61 62

64 65 66 67 68 69 70

72 73 74 75 76 77 78

79 80 81 82 83 84 85

86 87 88 89 90 91 92

94 95 96 97 98 99 100

102 103 104 105 106 107 108

109 110 111 112 113 114 115

116 117 118 119 120 121 122

124 125 126 127 128 129 130

132 133 134 135 136 137 138

140 141 142 143 144 145 146

148 149 150 151 152 153 154

156 157 158 159 160 161 162

164 165 166 167 168 169 170

172 173 174 175 176 177 178

180 181 182 183 184 185 186

188 189 190 191 192 193 194

196 197 198 199 200 201 202

204 205 206 207 208 209 210

212 213 214 215 216 217 218

219 220 221 222 223 224 225

227 228 229 230 231 232 233

235 236 237 238 239 240 241

247 248 249 250 251 252 253

256 257 258 259 260 261 262

264 265 266 267 268 269 270

282 283 284 285 286 287 288

290 291 292 293 294 295 296

298 299 300 301 302 303 304

306 307 308 309 310 311 312

318 319 320 321 322 323 324

362 363 364 365 366 367 368

376 377 378 379 380 381 382

391 392 393 394 395 396 397

399 400 401 402 403 404 405

411 412 413 414 415 416 417

426 427 428 429 430 431 432

451 452 453 454 455 456 457

466 467 468 469 470 471 472

481 482 483 484 485 486 487

496 497 498 499 500 501 502

508 509 510 511 512 513 514

522 523 524 525 526 527 528

536 537 538 539 540 541 542

551 552 553 554 555 556 557

566 567 568 569 570 571 572

581 582 583 584 585 586 587

596 597 598 599 600 601 602

611 612 613 614 615 616 617

626 627 628 629 630 631 632

641 642 643 644 645 646 647

656 657 658 659 660 661 662

671 672 673 674 675 676 677

686 687 688 689 690 691 692

696 697 698 699 700 701 702

711 712 713 714 715 716 717

726 727 728 729 730 731 732

741 742 743 744 745 746 747

756 757 758 759 760 761 762

771 772 773 774 775 776 777

786 787 788 789 790 791 792

796 797 798 799 800 801 802

811 812 813 814 815 816 817

826 827 828 829 830 831 832

841 842 843 844 845 846 847

856 857 858 859 860 861 862

871 872 873 874 875 876 877

886 887 888 889 890 891 892

896 897 898 899 900 901 902

911 912 913 914 915 916 917

926 927 928 929 930 931 932

941 942 943 944 945 946 947

956 957 958 959 960 961 962

971 972 973 974 975 976 977

986 987 988 989 990 991 992

996 997 998 999 1000 1001 1002

1011 1012 1013 1014 1015 1016 1017

1026 1027 1028 1029 1030 1031 1032

1041 1042 1043 1044 1045 1046 1047

1056 1057 1058 1059 1060 1061 1062

1071 1072 1073 1074 1075 1076 1077

1086 1087 1088 1089 1090 1091 1092

1096 1097 1098 1099 1100 1101 1102

1111 1112 1113 1114 1115 1116 1117

1126 1127 1128 1129 1130 1131 1132

1141 1142 1143 1144 1145 1146 1147

1156 1157 1158 1159 1160 1161 1162

1171 1172 1173 1174 1175 1176 1177

1186 1187 1188 1189 1190 1191 1192

1196 1197 1198 1199 1200 1201 1202

1211 1212 1213 1214 1215 1216 1217

1226 1227 1228 1229 1230 1231 1232

1241 1242 1243 1244 1245 1246 1247

1256 1257 1258 1259 1260 1261 1262

1271 1272 1273 1274 1275 1276 1277

1286 1287 1288 1289 1290 1291 1292

1296 1297 1298 1299 1300 1301 1302

1311 1312 1313 1314 1315 1316 1317

1326 1327 1328 1329 1330 1331 1332

1341 1342 1343 1344 1345 1346 1347

1356 1357 1358 1359 1360 1361 1362

1371 1372 1373 1374 1375 1376 1377

1386 1387 1388 1389 1390 1391 1392

1396 1397 1398 1399 1400 1401 1402

1411 1412 1413 1414 1415 1416 1417

1426 1427 1428 1429 1430 1431 1432

1441 1442 1443 1444 1445 1446 1447

1456 1457 1458 1459 1460 1461 1462

1471 1472 1473 1474 1475 1476 1477

1486 1487 1488 1489 1490 1491 1492

1496 1497 1498 1499 1500 1501 1502

1511 1512 1513 1514 1515 1516 1517

1526 1527 1528 1529 1530 1531 1532

1541 1542 1543 1544 1545 1546 1547

1556 1557 1558 1559 1560 1561 1562

1571 1572 1573 1574 1575 1576 1577

1586 1587 1588 1589 1590 1591 1592

1596 1597 1598 1599 1600 1601 1602

1611 1612 1613 1614 1615 1616 1617

1626 1627 1628 1629 1630 1631 1632

1641 1642 1643 1644 1645 1646 1647

1656 1657 1658 1659 1660 1661 1662

1671 1672 1673 1674 1675 1676 1677

1686 1687 1688 1689 1690 1691 1692

1696 1697 1698 1699 1700 1701 1702

1711 1712 1713 1714 1715 1716 1717

1726 1727 1728 1729 1730 1731 1732

1741 1742 1743 1744 1745 1746 1747

1756 1757 1758 1759 1760 1761 1762

1771 1772 1773 1774 1775 1776 1777

1786 1787 1788 1789 1790 1791 1792

1796 1797 1798 1799 1800 1801 1802

1811 1812 1813 1814 1815 1816 1817

1826 1827 1828 1829 1830 1831 1832

1841 1842 1843 1844 1845 1846 1847

1856 1857 1858 1859 1860 1861 1862

1871 1872 1873 1874 1875 1876 1877

1886 1887 1888 1889 1890 1891 1892

1896 1897 1898 1899 1900 1901 1902

1911 1912 1913 1914 1915 1916 1917

1926 1927 1928 1929 1930 1931 1932

1941 1942 1943 1944 1945 1946 1947

1956 1957 1958 1959 1960 1961 1962

1971 1972 1973 1974 1975 1976 1977

1986 1987 1988 1989 1990 1991 1992

1996 1997 1998 1999 2000 2001 2002

2011 2012 2013 2014 2015 2016 2017

2026 2027 2028 2029 2030 2031 2032

2041 2042 2043 2044 2045 2046 2047

2056 2057 2058 2059 2060 2061 2062

2071 2072 2073 2074 2075 2076 2077

2086 2087 2088 2089 2090 2091 2092

2096 2097 2098 2099 2100 2101 2102

2111 2112 2113 2114 2115 2116 2117

2126 2127 2128 2129 2130 2131 2132

2141 2142 2143 2144 2145 2146 2147

2156 2157 2158 2159 2160 2161 2162

2171 2172 2173 2174 2175 2176 2177

2186 2187 2188 2189 2190 2191 2192

2196 2197 2198 2199 2200 2201 2202

2211 2212 2213 2214 2215 2216 2217

2226 2227 2228 2229 2230 2231 2232

2241 2242 2243 2244 2245 2246 2247

2256 2257 2258 2259 2260 2261 2262

2271 2272 2273 2274 2275 2276 2277

2286 2287 2288 2289 2290 2291 2292

2296 2297 2298 2299 2300 2301 2302

2311 2312 2313 231

April 2016

18

Monday

Graphic Arts Institute

April 2016

Array Creation: One dimensional?

Every array in java is object only. Hence we can create arrays by using new operator.

`int[] a = new int[3];`



`s.o.p(a.get(0).get object
("name"));`

19

Tuesday `out: [I`

April 2016

For every array type corresponding classes are available and these classes are part of java language and not available ~~to~~ to the programmar level.

April 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	1					
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

April 2016

Graphic Arts Institute

20

Wednesday

Time

April 2016

Array type	corresponding class
int[]	[I]
int[][]	[[I]]
double[]	[D]
short[]	[S]
byte[]	[B]
boolean[]	[Z]

21

Thursday

Time

April 2016

~~Related~~ ① At the time of array creation compulsory we should specify the size - otherwise we will get compile time error.

int[] x = new int[]; X

int[] x = new int[6]; ✓

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

April 2016

22 Friday

Graphic Arts Institute



April 2016

② It is legal to have an array with size "0" in java.

int[] a = new int[0]; ✓

③ If we are trying to specify array size with some negative int value, then we will get runtime exception saying:

23 Saturday

April 2016

negativearraysize exception.

int[] a = new int[-6];

RE: negativearraysize

April 2016

Mon Tue Wed Thu Fri

1

4 5 6 7 8

11 12 13 14 15

18 19 20 21 22

25 26 27 28 29

first lesson

last lesson

April 2016

Graphic Arts Institute

24

Sunday

April 2016

① To specify array size that allowed data types are: byte, short, char, int.

If we are trying to specify any other type then we will get compile time error.

✓ `int[] x = new int[10];`

✓ `int[] x = new int['a'];`

25 Monday

April 2016

`byte b = 10;`

✓ `int[] x = new int[b];`

`short s = 20;`

✓ `int[] x = new int[s];`

✗ `int[] x = new int[10L];`

CE: P L P

found: long
required: int

April 2016
Mon Tue Wed Thu Fri
1
4 5 6 7 8
11 12 13 14 15
18 19 20 21 22
25 26 27 28 29

April 2016

26 Tuesday

Graphic Arts Institute

April 2016

byte → short

char

int → long → float → double

- ⑤ The maximum allowed array size in Java is 2147483649, which is the maximum value of int data type.

27 Wednesday

April 2016

✓ `int[] x = new int[2147483649];`

✗ `int[] x = new int[2147483648];`

CE: integer number too large

Even in the first case we make a runtime exception, if sufficient heap memory not available.

April 2016
Mon Tue Wed Thu Fri Sat Sun
10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

April 2016

Graphic Arts Institute

28 Thursday

April 2016

Two dimensional Array creation

In java two dimensional array not implemented by using matrix style. So, people followed array of arrays approach for multidimensional array creation. The main advantage of this approach is memory utilization will be improved.

29 Friday

April 2016

Ex:

`int [][] x = new int [2][];`

`x[0] = new int [2];`

`x[1] = new int [5];`

April 2016
Su Mo Tu We Th Fr Sa
30 1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

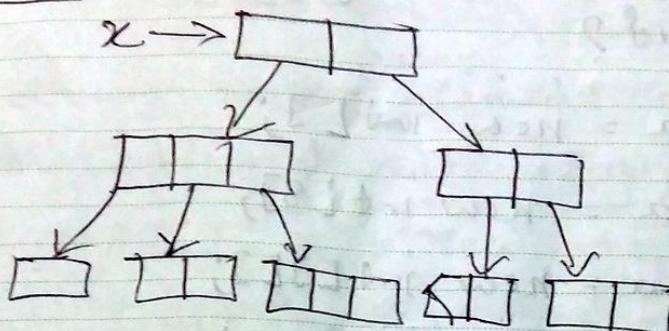
April 2016

6

30 Saturday

Graphic Arts Institute

April 2016

Ex 2:

01 Sunday

Time

May 2016

 $\text{int} \text{ } * \text{ } x = \text{new int}[2][2];$
 $x[0] = \text{new int}[3][];$
 $x[0][0] = \text{new int}[1];$
 $x[0][1] = \text{new int}[2];$
 $x[0][2] = \text{new int}[3];$
 $x[1] = \text{new int}[2][2]$

May 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

May 2016

Graphic Arts Institute

02 Monday

Time

May 2016

1. Which of the array declarations
are valid?

int[] a = new int[];

int[] a = new int[3];

int[][] a = new int[][];

int[][] a = new int[3][];

int[][] a = new int[][][9];

int[][] a = new int[3][4];

int[][] a = new int[3][4];

03 Tuesday

Time

May 2016

int[3][4][5] a = new int[3][4][5];

int[3][4][] a = new int[3][4][];

int[3][4][5] a = new int[3][4][5];

int[3][4][5] a = new int[3][4][5];

May 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	6
8	9	10	11	12	13
15	16	17	18	19	20
22	23	24	25	26	27
29	30	31			

May 2016

04

Wednesday

Graphic Arts Institute

May 2016

Array initialization:

Once we creates an array every element by default, initialized with default values.

```
int [] x = new int [2];
```

```
System.out.println(x); [I@123456
```

```
System.out.println(x[0]);
```

Output: 0

default value
initialization

$x \rightarrow [0 | 0]$

05

Thursday

May 2016

Whenever we are trying to print any reference variable, internally `toString()`, will be called, which is implemented by default to return the string in the following form.

May 2016

Su	Mo	Tu	We	Th	Fr
1	2	3	4	5	6
7	8	9	10	11	12
14	15	16	17	18	19
21	22	23	24	25	26
28	29	30	31		

May 2016

06 Friday

Graphic Arts Institute

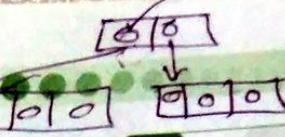
May 2016

int [] x = new int [2][3];

S. O. P (x); EI @ 32ef23

S. O. P (x[0]); EI @ 22ef24

S. O. P (x[0][0]); O



07 Saturday

May 2016

theory of point - m and associated
classmate addressed some for you
is Dr. Dr. Baloo and this, O friend,
what is related w/ estimation
not possible - it is present - it

May 2016

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29					

May 2016

08

Sunday

Graphic Arts Institute



May 2016

int [] [] x = new int [2] [] ;

s . o . p (x) ; E [I @ 2 f e c 1 2]

s . o . p (x [0]) ; null

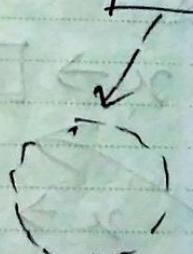
s . o . p (x [0] [0]) ; R . E . null pointer exception

09

Monday

May 2016

x → [null null]



If we are trying to perform any operation on null, then we will get runtime exception; trying null pointer exception.

May 2016

Mon Tue Wed Thu Fri
1 2 3
4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

May 2016

Graphic Arts Institute

10 Tuesday

Time

May 2016

Once we create an array, our array ~~has~~ elements by default ~~is~~ initialized with default values (0). If we are not satisfied with default values, then we override ~~with~~ these values with our customized values.

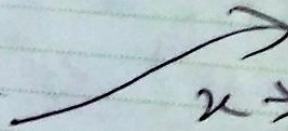
11 Wednesday

Time

May 2016

$$\text{int } x[2] \quad x = \text{new int}[2];$$


$$x \rightarrow \boxed{0} \boxed{0}$$

$$x[0] = 5;$$


$$x \rightarrow \boxed{5} \boxed{0}$$

$$x[1] = 6;$$

May 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

$$x[2] = 10; \quad \text{Array}$$

RE: Array Index Out of bound

$$x[-1] = 10; \rightarrow \text{exception}$$

May 2016

12 Thursday

Graphic Arts Institute



May 2016

 $x[1..2] = 5;$ CE: p2p

found: double

required: int

* We can declare, create and initialize an array in a single line (Shortcut Representation)

13 Friday

May 2016

~~int [] x; new int [3];~~

$x = \text{new int}[3];$

$x[0] = 10;$

$x[1] = 20;$

$x[2] = 30;$

$\rightarrow \text{int } x = \{10, 20, 30\}.$

$\text{char } ch = \{'a', 'e', 'b'\}$

$\text{String } s = \{"a", "aa", "ab"\}$

May 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	6
8	9	10	11	12	13
15	16	17	18	19	20
22	23	24	25	26	27
29	30	31			

May 2016

Graphic Arts Institute

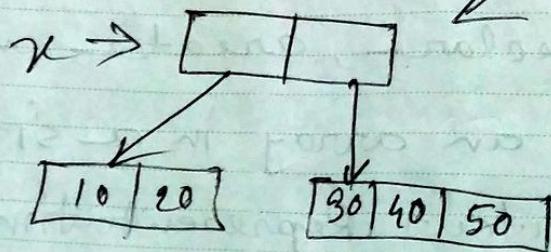
14

Saturday

Time

May 2016

We can use this shorthand for multidimensional arrays also.

 $\text{int } x = \{ \{10, 20\}, \{30, 40, 50\} \}$


15

Sunday

Time

May 2016

 $\text{int } x = \{ \{ \{10, 20, 30\}, \{40, 50, 60\} \}, \{ \{70, 80, 90, 100, 110\} \} \};$

S.O.P ($x[0][1][2]$); O.P: 60

S.O.P ($x[1][0][1]$); op: 80

S.O.P ($x[2][0][0]$); RE: AJOOBE

S.O.P ($x[1][2][0]$); RE: AJOOBE

S.O.P ($x[1][1][1]$); 100

S.O.P ($x[2][1][0]$); RE: AJOOBB

May 2016

1	2	3	4	5	6
7	8	9	10	11	
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29
30	31				

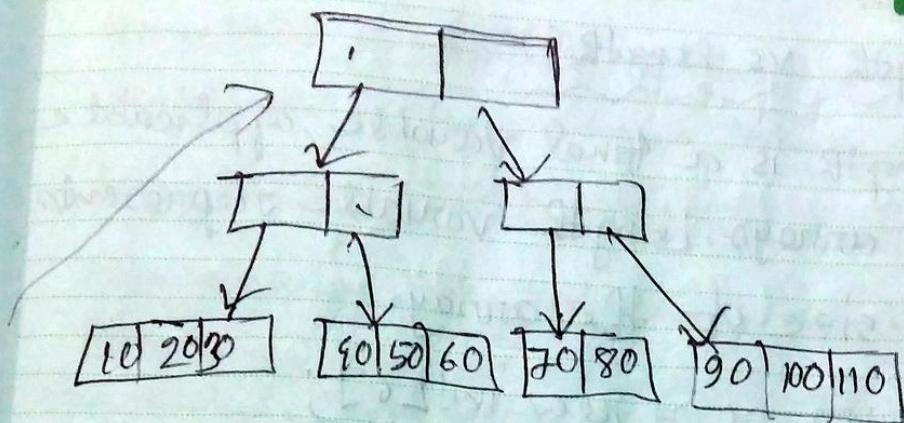
May 2016

16 Monday

Graphic Arts Institute



May 2016



* If we want to use this shortcut, compulsorily we should perform all activities in a single line. If we

17 Tuesday

Time

May 2016

are trying to divide into multiple line then, we will get compile time error.

int [] x = {10, 20, 30};

✓
int [] x;

May 2016

Sun Mon Tue Wed Thu Fri
1 2 3 4 5 6
8 9 10 11 12 13
15 16 17 18 19 20
22 23 24 25 26 27
29 30 31

CE : illegal start of Expression,

May 2016

Graphic Arts Institute

18

Wednesday

Time

May 2016

length vs length()

Length is a final variable, applicable for arrays. Length variable represents the size of the array.

int[] x = new int[6];

S.O.P(x.length);

S.O.P(x.length);

S.P: 6

May 2016

19

Thursday

Time

✓
CB: cannot find symbol
symbol: method length()
location: class int[]

May 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31					

May 2016



Graphic Arts Institute



20 Friday

Time

May 2016

length() method is a final method,
Applicable for String objects.

length() method returns number
of character present in the
string.

String s = "Hello";

s.o.p(s.length);

21 Saturday

Time

May 2016

s.o.p(s.length());

o.p: 5

CE: can not find symbol
symbol: variable length

Location: class java.lang.String

May 2016

Sat Sun Mon Tue Wed Thu Fri

Length variable applicable for arrays but not
for String objects, whereas length() ~~is a method~~
applicable for String object but not for
arrays.

May 2016

Graphic Arts Institute

May 2016

22

Sunday

Time

`String[] s = {"A", "AA", "AAA"};`

- ✓ ① `s.o.p(s.length);` 3 → CE: can not find symbol
symbol: method length
- ✗ ② `s.o.p(s.length());` → Method: class String
- ✗ ③ `s.o.p(s[0].length);` → CE: can not find symbol
symbol: variable len
length: class J.R. String
- ✓ ④ `s.o.p(s[0].length());` 1

23

Monday

Time

May 2016

* In multidimensional arrays, length variable represent only base size but not total size.

`int[][] x = new int[6][3];`

`s.o.p(x.length);` 6

`s.o.p(x[0].length);` 3

May 2016

Sat Sun Mon Tue Wed Thu Fri

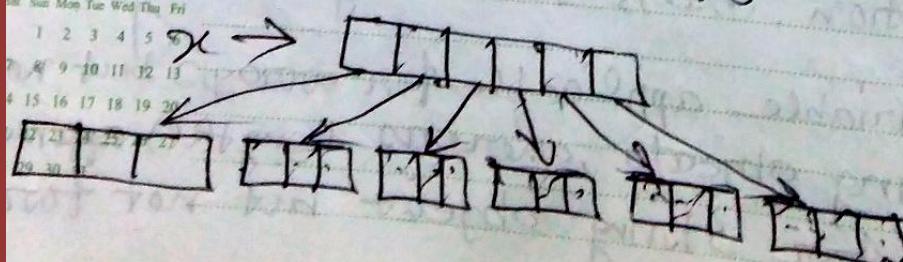
1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26

27 28 29 30 31



May 2016

24 Tuesday

Graphic Arts Institute



May 2016

There is no direct way to specify total length of multidimensional array. But indirectly we can find as follows.

$x[0].length + x[1].length + \dots + x[5].length$

* Anonymous Array: Sometimes we can declare an array without name, such type of nameless arrays are called anonymous arrays.

25

Wednesday

Time

May 2016

The main purpose of anonymous array is just for instant use (one time use).

We can create anonymous arrays as follows:

`new int[] {10, 20, 30, 40}` ✓

May 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

while creating anonymous arrays we can't specify the size, otherwise we will get compile time error:

`new int[3] {10, 20, 30} X, new int[] {10, 20, 30} Y`

May 2016

Graphic Arts Institute

May 2016

26

Thursday

Time

we can create multi-dimensional arrays also!

new int[][] {{10, 20}, {30, 40, 50}};

Based on our requirement we can give the name for anonymous array thus it is no longer anonymous.

✓ int[] x = new int[] {10, 20, 30};

27

Friday

Time

May 2016

Array element assignments:

Case 1:

In the case of primitive type arrays, as array elements we can provide any type which can be implicitly promoted to declare these type.

May 2016

Sat Sun Mon Tue Wed Thu Fri
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

✓ $x[0] = 10;$

short a = 30;

✓ $x[1] = 'a';$

✓ $x[2] = D;$

byte b = 20;
✓ $x[3] = b;$

✓ $x[4] = 10.5;$

CB: PLP
bound: long
datatype: int

May 2016

28 Saturday

Time

Graphic Arts Institute

May 2016

Case 2:

In the case of object type arrays, as array elements, we can provide, either declared type object or its child class object.

Ex1:

Object[] a = new Object[10];

a[0] = new Object(); ✓

a[1] = new String("durga"); ✓

a[2] = new Integer(10); ✓

29 Sunday

Time

May 2016

Ex 2:

Number[] n = new Number[10];

✓ n[0] = new Integer(10);

✓ n[1] = new Double(10.5);

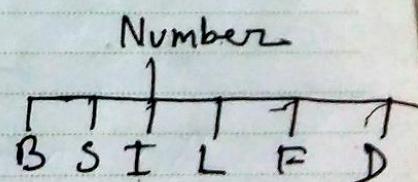
X n[2] = new String("durga");

May 2016

CE: incompatible types

found: J. L. String

required: J. L. Number



Sat Sun Mon Tue Wed Thu Fri

1 2 3 4

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31

May 2016

Graphic Arts Institute

30 Monday

May 2016

Case - 3:

For interface type arrays, as array elements, its implementation class objects are allowed:

Runnable[] or = new Runnable[];

✓ arr[0] = new Thread();

X arr[1] = new String("surga");

Runnable(i)



Thread

31

Tuesday

May 2016

CE: incompatible types
found: java.lang.String
required: Runnable

May 2016

Mon	Tue	Wed	Thu	Fri
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31				

June 2016



Graphic Arts Institute



01

Wednesday

Time

June 2016

Array type

primitive → arrays

Allowed element type

Any types which can be implicitly promoted to declared type.

Object type arrays

Either declared type or its child class objects.

Abstract class type arrays

Its child class objects.

02

Thursday

Time

June 2016

interface type arrays

Its implementation class objects are allowed.

June 2016

Sun Mon Tue Wed Thu Fri Sat

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30

June 2016

Graphic Arts Institute

03 Friday

Time

June 2016

Array variable assignment:

Case 1: Element level promotions are not applicable at array level. For example char element can be promoted to int type whereas char array cannot be promoted to int array.

```
int[] x = {10, 20, 30};
```

```
char[] ch = {'a', 'b', 'c'};
```

04 Saturday

Time

June 2016

✓ $\text{int}[] b = x;$ $\text{char} \rightarrow \text{int}$ ✓

✗ $\text{int}[] c = ch;$ $\text{char}[]$ $\text{int}[]$
 ↴
 CE: incompatible types ↴ ↴

found: $\text{char}[]$

required: $\text{int}[]$

June 2016

Mon Tue Wed Thu Fri

1 2 3

4 5 6 7 8 9 10

13 14 15 16 17

20 21 22 23 24

27 28 29 30

June 2016

05 Sunday

Graphic Arts Institute

June 2016

which of the following promotions are perform automatically:

- ✓ char → int
- ✗ char [] → int []
- ✓ int → double
- ✗ int [] → double []
- ✗ float → int
- ✗ float [] → int []

06 Monday

June 2016

✓ String → Object

✓ String [] → Object []

But in the case of Object type arrays, child class type array can be promoted to parent class type array.

Ex: { String s = {"A", "AB", "AAA"};
Object [] o = s; }

June 2016
Sat Sun Mon Tue Wed Thu Fri
1 2
3 4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30

June 2016

June 2016

07 Tuesday

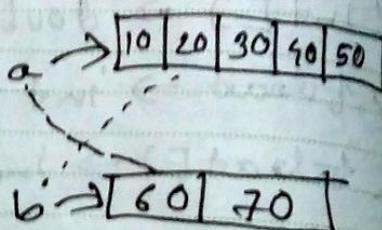
Time

Case-2: Whenever we are assigning to one array to another array internal element won't be copied, just reference variable will be reassigned.

$$\text{int } [] \ a = \{10, 20, 30, 40, 50\}$$

$$\text{int } [] \ b = \{60, 70\}$$

$$\checkmark 1. \ a = b$$

$$\checkmark 2. \ b = a$$


08 Wednesday

Time

June 2016

Case-3: Whenever we are assigning one array to another array, ~~internal elements~~ the dimension must be matched.

Eg: In the place of one dimensional $\text{int } []$ we should provide one dimensional array only.

If we are trying to provide any other dimension, then we will get compile time error.

 6 7 8 9 10
 11 12 13 14 15
 16 17 18 19 20
 21 22 23 24 25
 26 27 28 29 30



June 2016

`int a[3][2] a = new int[3][2];`

~~`X a[0] = new int[4][2];`~~ → CE: incompatible types
 found: int[2]
 required: int[]

~~`X a[0] = new int[10];`~~ → CE: incompatible type
 found: int
 required: int[]

10 Friday



$\checkmark a[0] = \text{new int [2]};$

- Whenever we are assigning one array to another array, both dimension and type must be matched, but sizes are not required to match.

June 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3

89-16

11 12 13 14 15

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30

June 2016

Graphic Arts Institute



11

Saturday

Time

June 2016

*Types of Variables:

Based on type of value represented by a variable, all variables are divided into two types:

1. Primitive Variables : Can be used

To represent primitive values.

Ex: int x = 10;

12

Sunday

Time

June 2016

2. Reference Variables:

Can be used to refer objects.

Ex:

Student x = new Student();

$x \rightarrow$ Object
new Student()

June 2016

Mon Tue Wed Thu Fri

1 2 3

7 8 9 10

14 15 16 17

21 22 23 24

28 29 30

June 2016

13

Monday

Graphic Arts Institute



June 2016

Division Two:

Based on position of declaration and behaviour all variables are divided into three types:

- ① Instance Variable
- ② Static Variable
- ③ Local variables

14

Tuesday

Time

June 2016

Instance Variable: If the value of a variable is varied from object to object, such type of variables are called Instance variables.

For every object a separate copy of instance variable will be created.

June 2016
Sat Sun Mon Tue Wed Thu

1 2 3
4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30

15

Wednesday

June 2016

- * Instance variable should be declared within the class directly but outside of any method or block or constructor.
- * Instance variable will be created at the time of object creation and destroyed at the time of object destruction. Hence the scope of instance variable is

16

Thursday

June 2016

exactly same as scope of object.

- * Instance variable will be stored in the heap memory as the part of object.

June 2016
Mon Tue Wed Thu Fri
1 2 3
6 7 8 9 10
13 14 15 16 17
20 21 22 23 24
27 28 29 30

strongly binds two objects of the same class together

• between

June 2016

17 Friday

Graphic Arts Institute

June 2016

- * We can't access instance variable directly from static area, but we can access by using object reference.
- But we can access instance variable directly from ~~the~~ instance area.
- * For instance variable jvm will always provide default values and we are not required to perform initialization explicitly.

18 Saturday

June 2016

we are not required to perform initialization explicitly.

class Test {

int x;

double d;

boolean b;

String s;

P.S. V. main (String[] args) {

 Test t = new Test();

 System.out.println(t.x); → 0

 System.out.println(t.d); → 0.0

 System.out.println(t.b); → false

} System.out.println(t.s); → null

June 2016

June 2016

19

Sunday

Time

* Instance variable also known as Object level variables or attributes.

Static Variable:

If the value of a variable is not varied from object to object then it is not recommended to declare variable as instance variable.

20

Monday

Time

June 2016

We have to declare such type of variable as class level by using static modifier.

* In the case of instance variables for every object a separate copy will be created but in the case of static variable a single copy will be created at class level and shared by every object of the class.

June 2016

21

Tuesday

Graphic Arts Institute

June 2016

- * Static variable should be declared with in the class directly but outside of any method or block or constructor with static keyword.
- * Static variable will be created at the time of class loading and destroyed at the time of class unloading, hence the scope of static variable is exactly same as scope of dot class file.

22

Wednesday

June 2016

June 2016

Sun Mon Tue Wed Thu Fri

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30

June 2016

June 2016

23

Thursday

Time

java test

1. Start JVM
2. Create and Start Main Thread
3. Locate test.class file

Static Variable
creation

4. Load test.class

5. Execute Main method

Static Variable
destruction

6. Unload test.class

7. Terminate main thread

8. Shut Down JVM

24

Friday

Time

June 2016

Static variable will be stored in method area.

We can access static variable either by object variable or class name. But recommended to use by class Name. Within the same class it is not required to use class name and we can access directly.

June 2016

Mon Tue Wed Thu Fri
2 3
6 7 10
13 14 15 16 17
20 21
27 28 29 30

June 2016

25

Saturday

Graphic Arts Institute



June 2016

class Test {

 static int x = 10;

 public static void main(String[] args){

 Test t = new Test();

 S.O.P(t.x); ✓

 S.O.P(Test.x); ✓

 S.O.P(x); ✓

26

Sunday

June 2016

We can access static variable directly from both instance and static areas.

class Test {

 static int x = 10;

 P.S.V.main(String[] args){

 S.O.P(x); ✓

 }

 public void m1(){

 S.O.P(x); ✓

June 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3				
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

June 2016

June 2016

27 Monday

- * For static variables JVM will provide default values and we are not required to perform initialization explicitly.

```
class Test {
    static int a;
    static double d;
    static String s;
```

28 Tuesday

June 2016

```
P: S. V. main(String[] args){
    S. O. P(a); → 0
    S. O. P(d); → 0.0 } output
    S. O. P(s); → null }
```

Static variable also known as class level variable or fields

June 2016

6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30

June 2016

29

Wednesday

Graphic Arts Institute



June 2016

class Test {

static int x = 10;

int y = 20;

P. S. V. main(String[] args) {

Test t₁ = new Test(); → t₁t₁.x = 888;t₁.y = 999;Test t₂ = new Test(); → t₂S.O. P(t₂.x + "..." + t₂.y);

30

Thursday

}

O/P: 888... 20

June 2016

slides, borders and new addition logo

→ units of should be printed as

matrices should → 10, +1 border as

as files for print jobs, etc.

June 2016

Sun Mon Tue Wed Thu Fri

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30

July 2016

July 2016

01

Friday

Time

Local Variables: Sometimes to meet temporary requirements of the program, we can declare variable inside a method or block or constructor, such type of variables are called local or temporary or stack or automatic variables.

Local variable will be stored inside stack memory.

02

Saturday

Time

July 2016

Local variable will be created, while executing the block in which we declared it, once block execution completes, automatically ^{local variable} will be destroyed. Hence the scope of local variable is the block in which we declared it.

July 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

July 2016

03 Sunday

Graphic Arts Institute



July 2016

class Test {

p. s. v. main (String [] args) {

int i = 0;

for (int j = 0; j < 3; j++) {

i = i + j;

}

s. o. println (i + " " + j);

}

}

CE: Can not find symbol

Symbol: variable j

location: class Test

04 Monday

July 2016

class Test {

p. s. v. main (String [] args) {

try {

int j = Integer.parseInt ("ten");

} catch (NumberFormatException e) {

j = 10;

July 2016

Sat Sun Mon Tue Wed Thu Fri

30 31

1

2 3 4 5

s. o. p (j);

6 7 8 9 10 11 12

13 14 15

16 17 18 19

20 21 22

23 24 25 26 27 28 29

30

July 2016

July 2016

05 Tuesday

Time

* For local variables JVM won't provide default values, compulsory we should perform initialization explicitly, before using that variable. That is if we are not using, then it's not required to perform initialization.

06 Wednesday

July 2016

class Test {

p. s. v. main(String[] args) {

int x;

s. o. p("Hello");

}

class Test {

p. s. v. main(String[] args) {

int x;

}

s. o. p(x);

}

g)

CB: variable x might
not have been initialization

July 2016

07 Thursday

Graphic Arts Institute

July 2016

The only applicable modifier for local variable is final. By mistake if we are trying to apply any other modifier than we will get compile time error.

If we are not declaring with any modifier, then by default it is default but this rule

08 Friday

July 2016

is applicable only for instance and static variables but not for local variables.

Instance and static variables can be accessed by multiple threads simultaneously and hence these are

not thread safe. But for local variables, for every ~~these~~ thread a separate copy will be created, that's why local variables are thread safe.

July 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

July 2016

09

Saturday

Time

July 2016

local variable:

① int [] x;

~~s.o.p(x);~~ } \rightarrow cc: variable x
~~s.o.p(x[0]);~~ } might not have been
 initialized

② int [] x = new int [3];

s.o.p(x); 11@1256 ✓

s.o.p(x[0]); 0 ✓

10

Sunday

Time

July 2016

Once we creates an array, every array elements by default initialized with default values. Whether it is instance, static, or local array.

July 2016

Mon Tue Wed Thu Fri

4	5	6	7	8
11	12	13	14	15
8	19	20	21	22
5	26	27	28	29

July 2016

11 Monday

Graphic Arts Institute

July 2016

Var-Arc Methods (Variable number of argument methods):

Until 1.4v we can not declare a method with variable number of arguments. If there is a change in number of arguments compulsory we should go for new method. It increases length of the code and reduces readability.

12 Tuesday

July 2016

To ~~over~~ over comes this problem sun people introduces var-arg methods in 1.5 v. According to this we can declare a method which can take variable number of arguments such type of methods are call var-~~arg~~ arg methods.

July 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31		1			
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

July 2016

Graphic Arts Institute

July 2016

13

Wednesday

Time

$m1(int \dots x)$

we can declare a var-arg method
as above.

We can call this method by passing
any number of int values, including
0 number.

$m1();$ ✓ $m1(10, 20);$ ✓

$m1(10);$ ✓

14

Thursday

Time

July 2016

public static void $\& m1(int \dots x) \{$
S.O.F ("var-arg");
}

public static void main(String[] args) {

$m1();$ ✓

$m1(10);$ ✓

$m1(10, 20);$ ✓

July 2016

Mon Tue Wed Thu Fri

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

July 2016

15 Friday

Graphic Arts Institute



July 2016

Internally var-arg parameter will be converted into one dimensional array. Hence within var-arg method we can differentiate values by using index.

```
class Test {
```

```
public static void main(String[] args) {
```

```
    sum();
```

16 Saturday

```
    sum(10, 20);
```

Time

July 2016

```
    sum(10, 20, 30);
```

```
    sum(10, 20, 30, 40);
```

```
}
```

```
    public static void sum(int ... x) {
```

```
        int total = 0;
```

```
        for (int x1 : x) {
```

```
            total = total + x1;
```

```
}
```

```
    System.out.println("total:" + total);
```

July 2016

Sun Mon Tue Wed Thu Fri

21 1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

```
}
```

July 2016

17

Sunday

Time

July 2016

$m1(\text{int } x, \text{ int... } y)$
 $\checkmark m1(\text{String } s, \text{ double... } y)$

$m1(\text{double... } d, \text{ String } s) \times$

$m1(\text{char } ch, \text{ String... } s) \checkmark$

Var-arg parameter should be last parameter.

18

Monday

Time

July 2016

$m1(\text{int... } x, \text{ double... } d) \times$

inside var-arg method we can take only one var-arg parameter and we can't take more than one var-arg parameter.

July 2016

Sun	Mon	Tue	Wed	Thu	Fri
30	31		1		
2	3	4	5	6	7
9	10	11	12	13	14
16	17	18	19	20	21
23	24	25	26	27	28
30					

July 2016

19

Tuesday

Graphic Arts Institute

July 2016

main() method:

Whether class contain main method or not, and whether main method is declared according to requirement or not, these things won't be checked by compiler. At runtime JVM is responsible to check these things.

20

Wednesday

July 2016

If JVM unable to find main method, then we will get runtime exception said:

No such method error: main
class Test{

July 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31		1			
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

}
java & Test.java ✓

Java Test → RE: No such Method Error:
main

July 2016

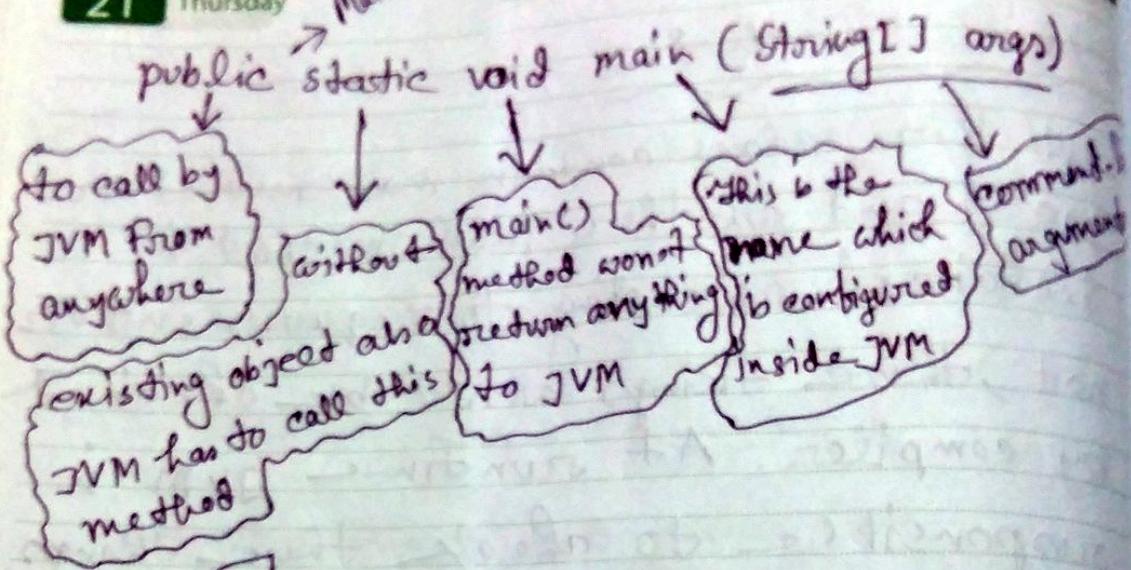
Graphic Arts Institute

July 2016

21

Thursday

Main Method & Syntax



22

Friday

Time

July 2016

At runtime JVM always searches for the main() method with the following prototype:

July 2016

Sun	Mon	Tue	Wed	Thu	Fri
31	1				
2	3	4	5	6	7
8	9	10	11	12	13
14	15	16	17	18	19
20	21	22	23	24	25
26	27	28	29		

1 Day

July 2016



Graphic Arts Institute



23

Saturday

Time

July 2016

- ① static public ✓
- ② main (String[] args) ✓
main (String []args) ✓
main (String args[]) ✓
- ③ main (String[] abc) ✓
- ④ main (String... args) ✓

24

Sunday

Time

July 2016

Even though above syntax is very strict, the following changes are acceptable!)

- ① instead of public static we can take static public, that is the order of modifier is not important.

July 2016
Sat Sun Mon Tue Wed Thu Fri
20 21 22 23 24 25 26
27 28 29 30 31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29

July 2016

July 2016

25 Monday

Time

- (i) we can declare string array in any acceptable form.
- (ii) Instead of args, we can take any valid java identifier.
- (iii) We can replace string[] with vararg parameters.

26 Tuesday

Time

July 2016

We can declare main method with the following modifiers:

- ✓ final
- ✓ synchronized
- ✓ strictfp

static final synchronized strictfp public
void main (String... args) {
} System.out.println("valid");

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

July 2016

27 Wednesday

Time

Graphic Arts Institute



July 2016

case I: Overloading of the main method is possible, but JVM will always call String[] args main method only.

The other overloaded method we have to call explicitly, like normal method call.

28 Thursday

Time

July 2016

class Test {

 public static void main(String[] args)

 {
 System.out.println("String");
 }

 public static void main(Integer[] args)

 {
 System.out.println("int[]");
 }

OP: String

Overloaded
methods

July 2016

Su	Mo	Tu	We	Th	Fr	Sa
30	31	1				
2	3	4	5	6	7	
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

July 2016

29 Friday

July 2016

case 2:

```

class P {
    public static void main(String[] args) {
        System.out.println("Parent main");
    }
}

```

~~public static void main()~~

30 Saturday

July 2016

class C extends P {

P.class

C.class

Java P ←

O/P: Parent main

Java C ← 1

Q/P: Parent main

Sun Mon Tue Wed Thu Fri

30 31

1 2 3 4 5 6 7

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

July 2016

31 Sunday

Graphic Arts Institute



July 2016

Inheritance concepts applicable for main method, hence while executing child class, if child doesn't contain main method, then parent class main method will be executed.

01 Monday

Time

August 2016

July 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
30	31		1			
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

August 2016

Graphic Arts Institute

02 Tuesday

Time

August 2016

Case 3:

class P {

| p s v main (String[] args) {
| s. o. p ("Parent main");
| }]

class C extends P {

| p. s v main (String[] args) {

03 Wednesday

Time

August 2016

| s. o. p ("child main");
| }]

Java c P. Java
↓
P. class C. class

August 2016

Java P <

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29 30 31

O/P: "Parent main"

Java C <

O/P: child main

August 2016

04 Thursday

Time

Graphic Arts Institute

August 2016

It seems overriding concept applicable
for main method, but it is not
overriding and it is method
hiding.

For main method inheritance and
overloading concepts are applicable
but overriding concept is not
applicable.

05 Friday

Time

August 2016

Instead of overriding method
hiding is applicable.

August 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29 30 31

August 2016

August 2016

06 Saturday

Time

class Test

3

1.6 V

javac Test.java ✓

Java Test

RE: NoSuchMethodError:
main1.7 V

javac Test.java ✓

Java Test

Error: Main method not found
in class Test, please declare
main method as:

public static void main(String[] args)

07 Sunday

Time

August 2016

August 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	
7	8	9	10	11	12
14	15	16	17	18	19
21	22	23	24	25	26
28	29	30	31		

August 2016

08 Monday

Graphic Arts Institute



August 2016

```
class Test {  
    static {  
        System.out.println("Static Block");  
    }  
}
```

Java 1.6 ✓

javac Test.java ✓

Java Test ↵

1.7 ✓

javac Test.java ✓

Java Test ↵

09 Tuesday

August 2016

O/p: static Block

Error: Main method not found in class Test.

R.E: NoSuchMethodError:
main

From 1.7v main method is mandatory,
to start program execution, hence
even though class contain main method,
Static block, it won't be executed
if the class doesn't contain main
method.

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5 6 7 8 9 10 11 12

13 14 15 16 17 18

19 20 21 22 23 24 25 26

27 28 29

August 2016

August 2016

10 Wednesday

Time

class Test {

static {

System.out.println("Static block");

System.exit(0);

}

{

1.6 ✓1.7 ✓

Java Test.java

Java Test.java ✓

Java Test

Java Test

O/P: Static Block

Error: Main method

August 2016

11

Thursday

Time

not found in class

Test .

August 2016

Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

30 31

August 2016

12

Friday

Time

Graphic Arts Institute



August 2016

class Test {

static {

System.out.print("Static Block");

}

~~System.out.println("Main Method");~~

P: S V main(String[] args) {

System.out.println("main method");

}

}

13

Saturday

Time

August 2016

1.6 V

Java Test.java

Java Test <

O/P: Static Block

Main Method

1.7 V

Java Test.java

Java Test <

O/P: Static Block

Main Method

August 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

August 2016

Graphic Arts Institute

14 Sunday

Time

August 2016

Command Line Arguments:

The arguments which are passing from command prompt are called command line arguments. With this command line arguments JVM will create an array and by passing that array as argument, JVM will call main method.

15 Monday

Time

August 2016

Java Test A B C

args[0]

args[1]

args[2]

args.length ≥ 3

August 2016

Sun Tue Wed Thu Fri

2 3 4 5

9 10 11 12

16 17 18 19

23 24 25 26

30 31

August 2016

16 Tuesday

Graphic Arts Institute



August 2016

Purpose of command line arguments:

The main objective of command line argument is, we can customize behaviour of the main method.

Case 1:

class Test {

17

Wednesday

August 2016

```
public static void main(String[] args) {
```

```
    for (int i = 0; i < args.length; i++) {
```

```
        System.out.println(args[i]);
```

```
}
```

```
{}
```

java Test A B C ↪ A B C RE: AIOBEK

August 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4

5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29 30 31

java Test ↪ RE: AIOBEK

A IOB → Array index out of Bound EK

August 2016

18 Thursday

Time

August 2016

case 2:

class Test {

public static void main(String[] args) {

String[] argh = {"x", "y", "z"};

args = argh;

for (String s : args) {

System.out.println(s);

}

19

Friday

Time

August 2016

}

java Test A B C ↴ x y z

args → [A B C]

argh → [x y z]

java Test A B ↴ x y z

java Test ↴ x y z

August 2016

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31				

August 2016

20 Saturday

Graphic Arts Institute



August 2016

case 3:

class Test {

 public static void main(String[] args) {

 System.out.println(args[0] + " " + args[1]);

}

}

Java Test 10 20 ↪ 10 20

Within main method command line

21 Sunday

August 2016

arguments are available in string form.

case 4: class Test {

 public static void main(String[] args) {

 System.out.println(args[0]);

}

Java Test "Note Book"
ops Note Book

August 2016

Sat Sun Mon Tue Wed Thu Fri

1 2 3 4 5

6 7 8 9 10 11 12

13 14 15 16 17 18 19

20 21 22 23 24 25 26

27 28 29 30 31

August 2016

August 2016

22

Monday

Time

~~# Java Coding Standards:~~

class A

{

public int m1(int x, int y)

{

return x+y;

}

}

23

Tuesday

Time

August 2016

package com.moudud.hassan

public class calculator

{

public static int add(int num1, int num2)

{

return num1+num2;

}

August 2016

Sat	Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

August 2016

24 Wednesday

Graphic Arts Institute



August 2016

Coding standard for classes:

string
stringBuffer
Account
Dog

Noun

25 Thursday

August 2016

Interface:

Runnable
Serializable
Comparable
RandomAccess

→ Adjective

August 2016

Su	Sun	Mon	Tue	Wed	Thu	Fri
	1	2	3	4	5	
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

August 2016

Graphic Arts Institute

26 Friday

Time

August 2016

* Methods:

print()
sleep()
run()

GetName
setSalary

→ Verbs

→ Verb-Noun combination

27 Saturday

Time

August 2016

* Variable:

name
age
salary
mobileNumber

→ Nouns

August 2016

Sun	Mon	Tue	Wed	Thu	Fri
1	2	3	4	5	
6	7	8	9	10	11
13	14	15	16	17	18
20	21	22	23	24	25
27	28	29	30	31	

August 2016

28

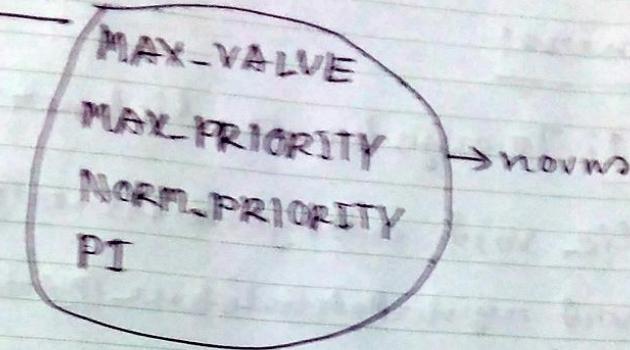
Sunday

Graphic Arts Institute



August 2016

constant:



These values are fixed, not changeable.

int
public static final MAX-VALUE = 5;

29

Monday

August 2016

Java Bean? → Simple Java class

Public class StudentBean

{

private String name;

public void setName(String Name)

{

this.name = name;

}

public String getName()

{

return name;

}

1

August 2016
Sat Sun Mon Tue Wed Thu Fri
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

August 2016

August 2016

30 Tuesday

Time

Listeners:case 1: To register a listener:

✓ public void addMyActionListener(MyActionListener)

✗ public void registerMyActionListener (MyActionListener)

✗ public void addMyActionListener (ActionListener l)

should be prefixed with "add" for method

31 Wednesday

Time

August 2016

case 2: To unregister a listener:

✓ public void removeMyActionListener(MyActionListener)

✗ public void unregisterMyActionListener (MyActionListener)

August 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5		
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Health care slide

September 2016

01

Thursday

Time

Graphic Arts Institute

September 2016

5

02

Friday

Time

September 2016

TO Be Continue _ _ _ _ _

September 2016

Sat Sun Mon Tue Wed Thu Fri

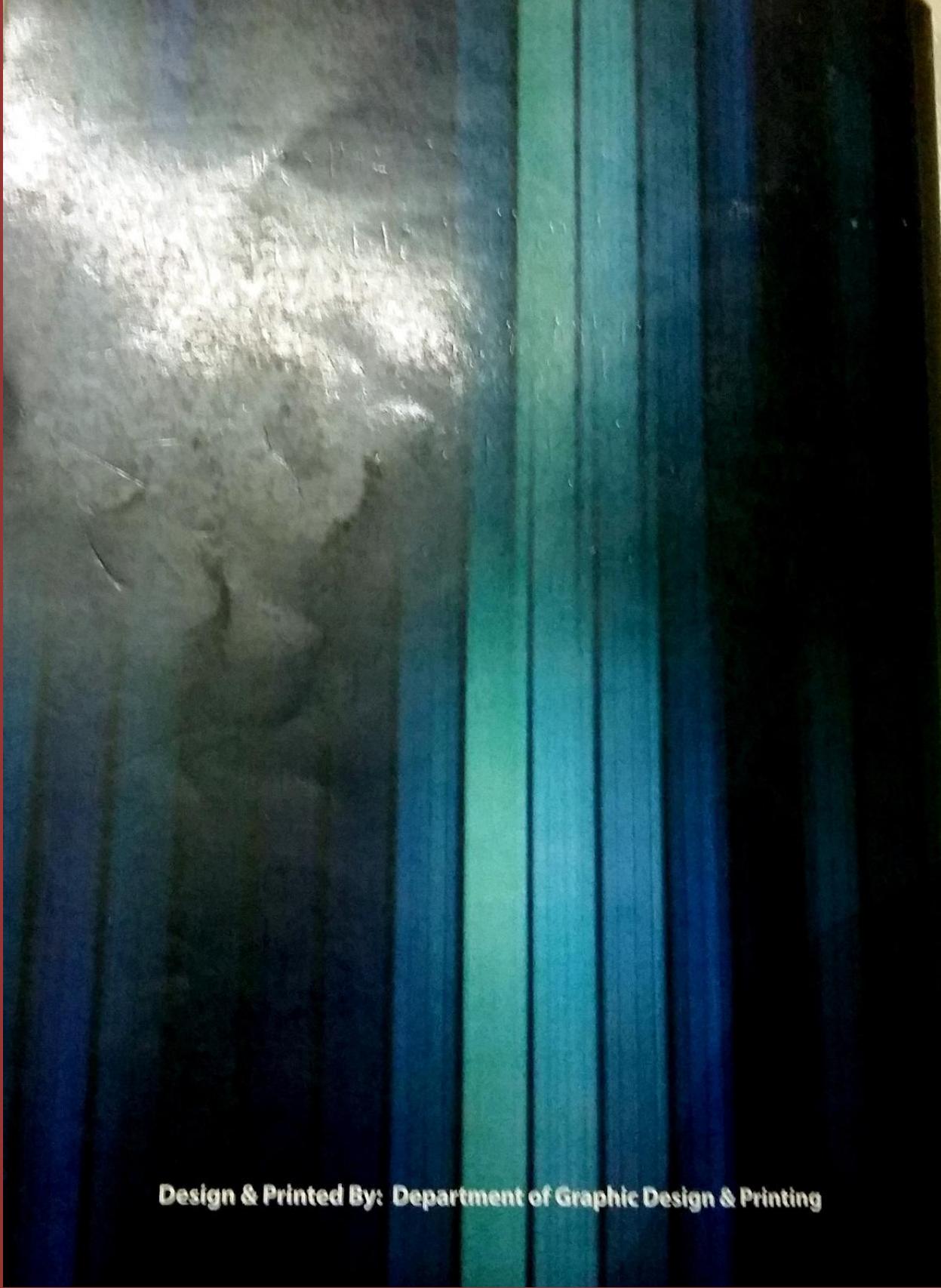
1 2

3 4 5 6 7 8 9

10 11 12 13 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30



Design & Printed By: Department of Graphic Design & Printing