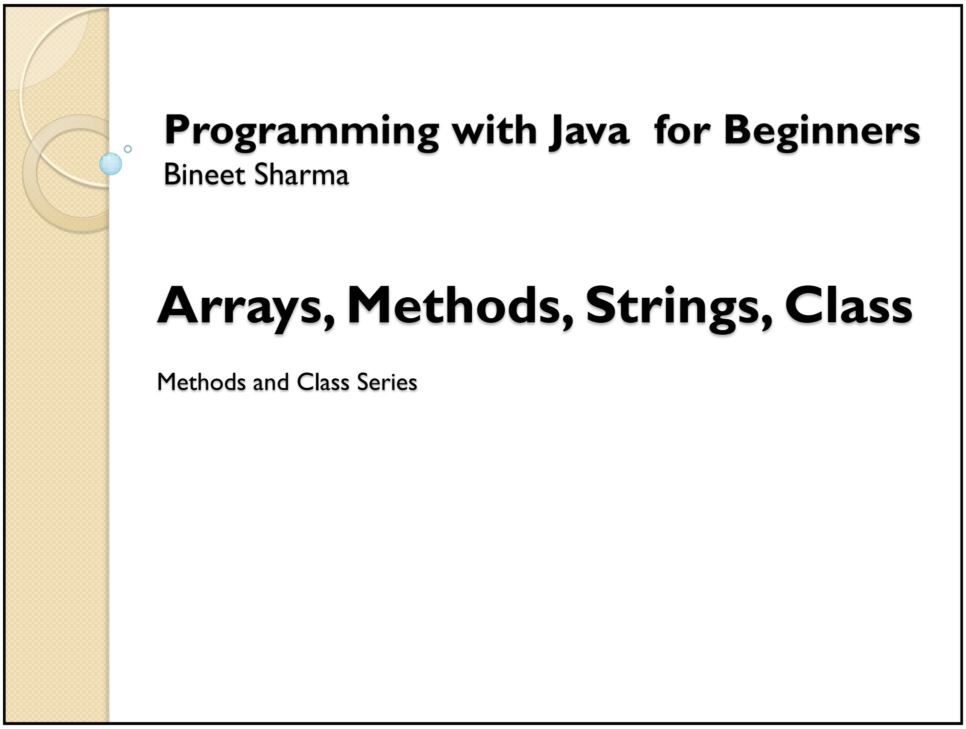
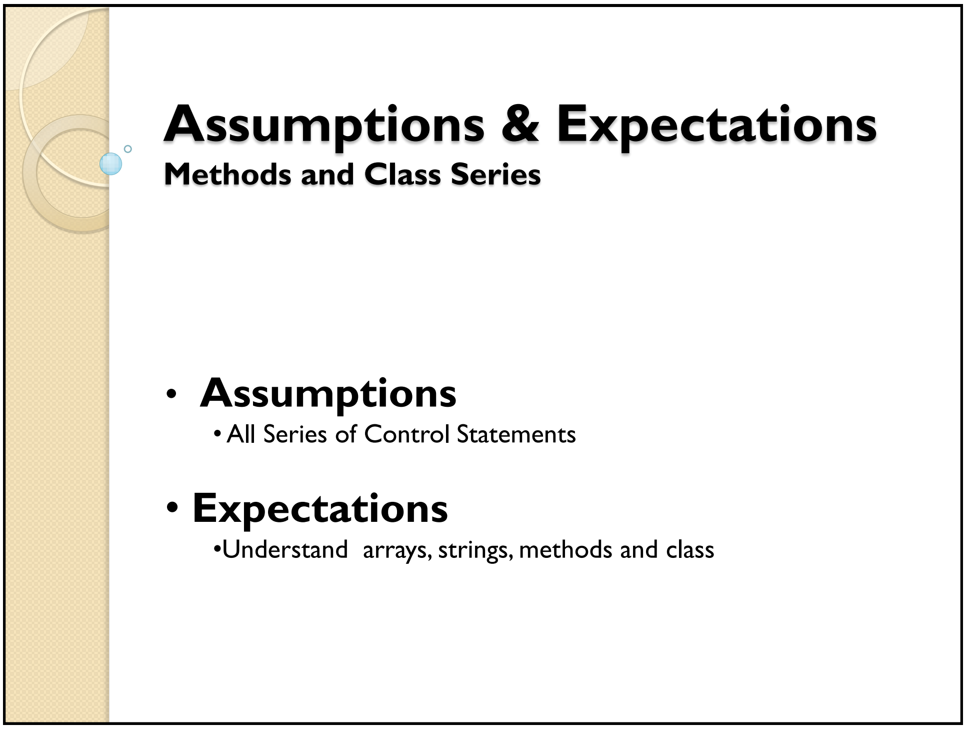


Programming with Java for Beginners

Bineet Sharma

Arrays, Methods, Strings, Class

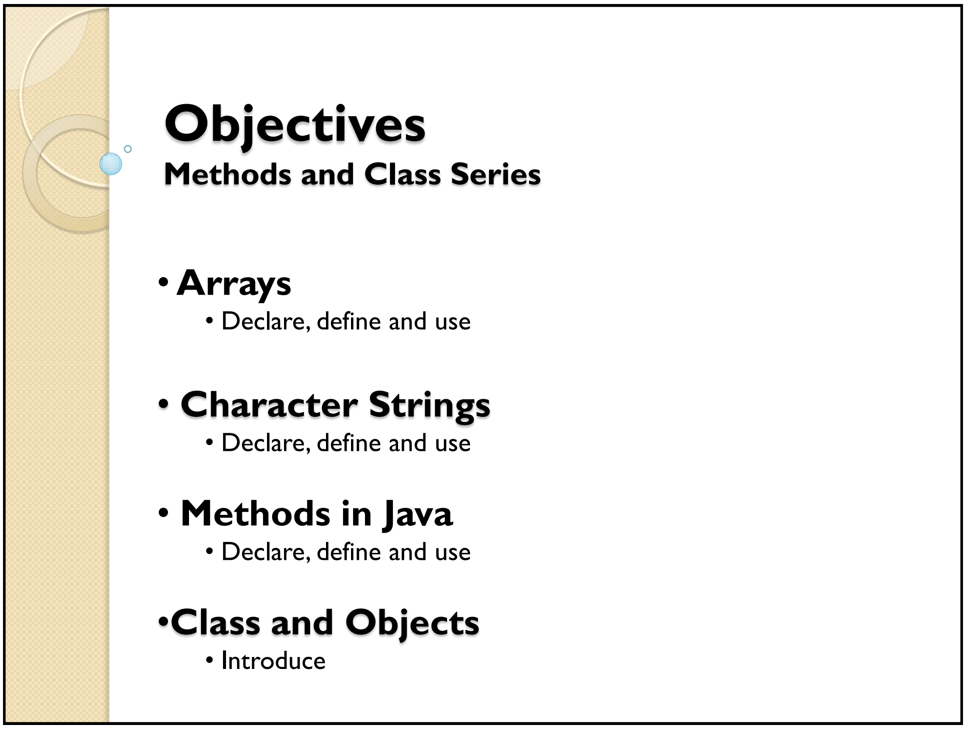
Methods and Class Series



Assumptions & Expectations

Methods and Class Series

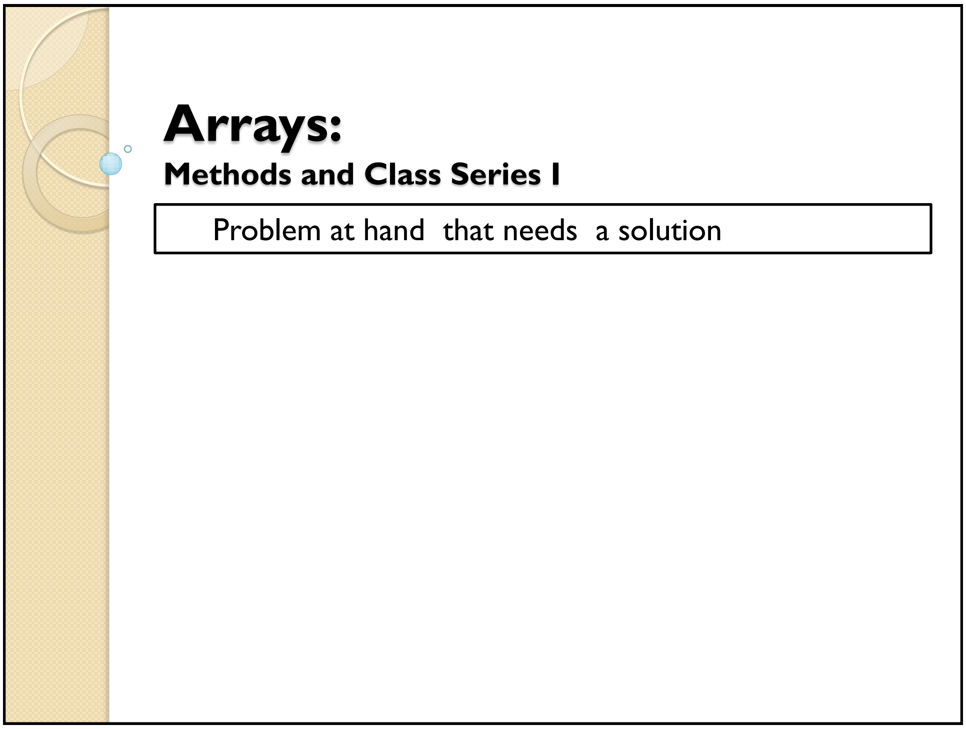
- **Assumptions**
  - All Series of Control Statements
- **Expectations**
  - Understand arrays, strings, methods and class



Objectives

Methods and Class Series

- **Arrays**
  - Declare, define and use
- **Character Strings**
  - Declare, define and use
- **Methods in Java**
  - Declare, define and use
- **Class and Objects**
  - Introduce



Arrays:

Methods and Class Series I

Problem at hand that needs a solution

Arrays:

Methods and Class Series I

Another problem at hand that needs a solution

Arrays:

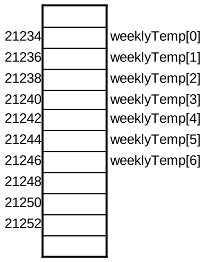
Methods and Class Series I

What is an array?

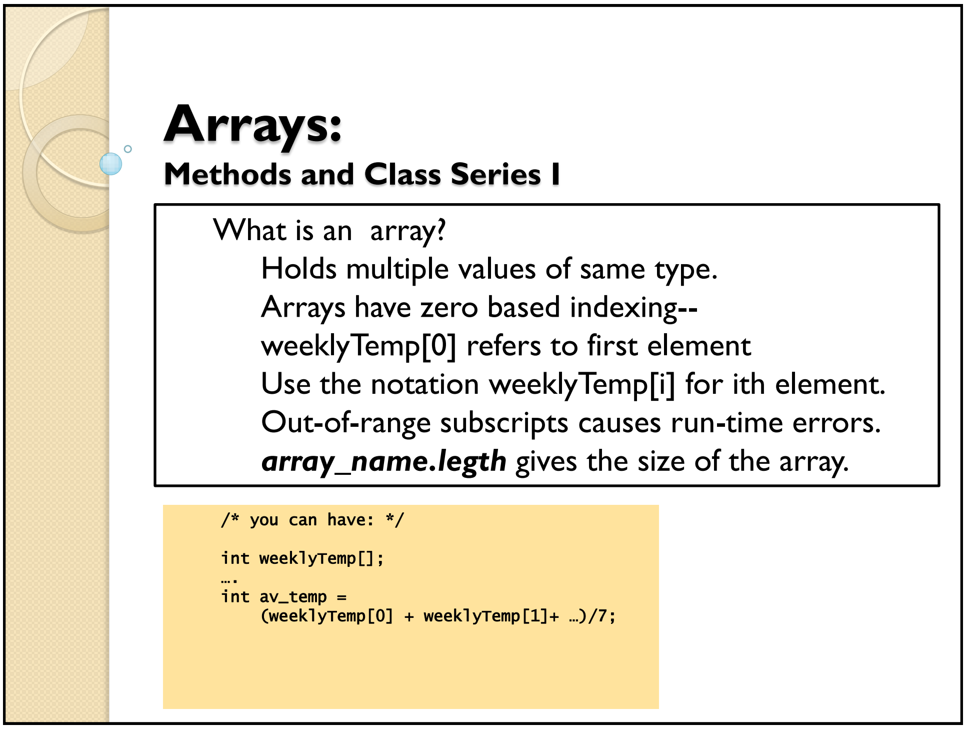
Holds multiple values of same type

```
/* Instead of: */

int sun_tmp;
int mon_tmp;
Int tue_tmp;
…
Int sat_tmp;

Int av_temp =
    (sun_tmp + mon_tmp + …)/7;
```

```
/* you can have: */

int weeklyTemp[];
….
int av_temp =
    (weeklyTemp[0] + weeklyTemp[1]+ …)/7;
```



21234	weeklyTemp[0]
21236	weeklyTemp[1]
21238	weeklyTemp[2]
21240	weeklyTemp[3]
21242	weeklyTemp[4]
21244	weeklyTemp[5]
21246	weeklyTemp[6]
21248	
21250	
21252	



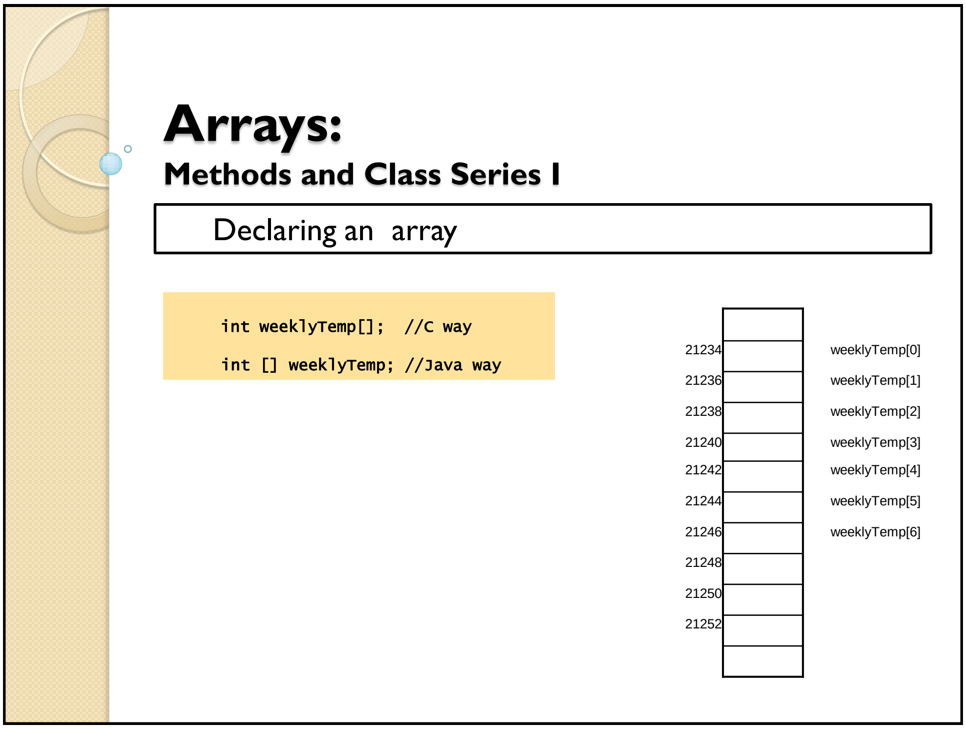
Arrays:

Methods and Class Series I

What is an array?

Holds multiple values of same type.
Arrays have zero based indexing--
weeklyTemp[0] refers to first element
Use the notation weeklyTemp[i] for ith element.
Out-of-range subscripts causes run-time errors.
array_name.legth gives the size of the array.

```
/* you can have: */

int weeklyTemp[];
….
int av_temp =
    (weeklyTemp[0] + weeklyTemp[1]+ …)/7;
```



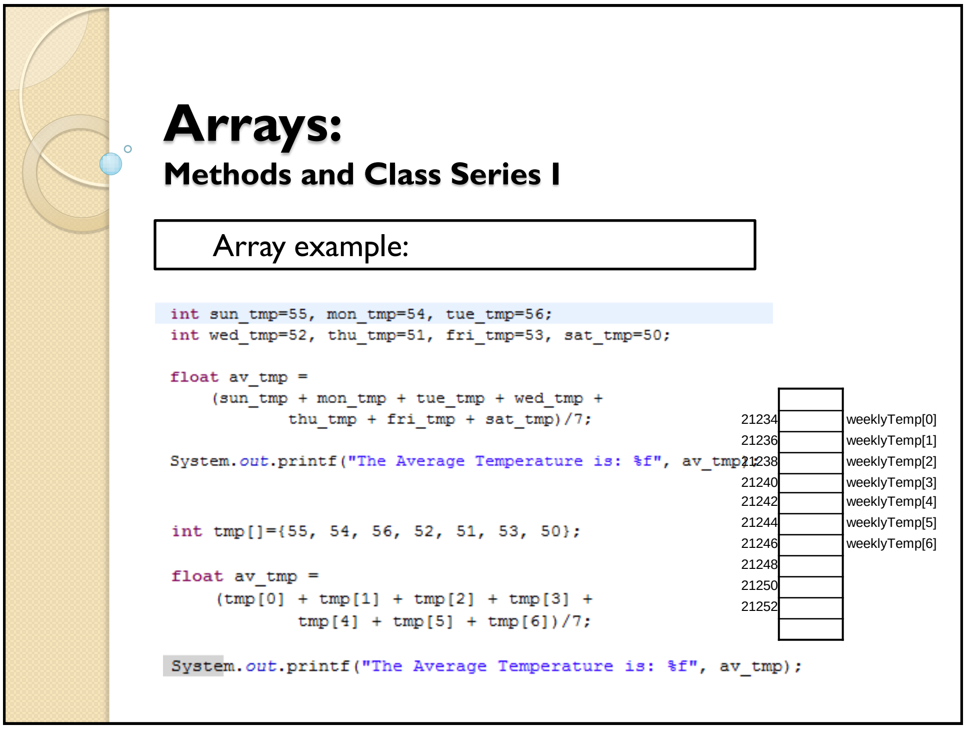
Arrays:

Methods and Class Series I

Declaring an array

```
int weeklyTemp[];  //C way

int [] weeklyTemp; //Java way
```

21234	weeklyTemp[0]
21236	weeklyTemp[1]
21238	weeklyTemp[2]
21240	weeklyTemp[3]
21242	weeklyTemp[4]
21244	weeklyTemp[5]
21246	weeklyTemp[6]
21248	
21250	
21252	



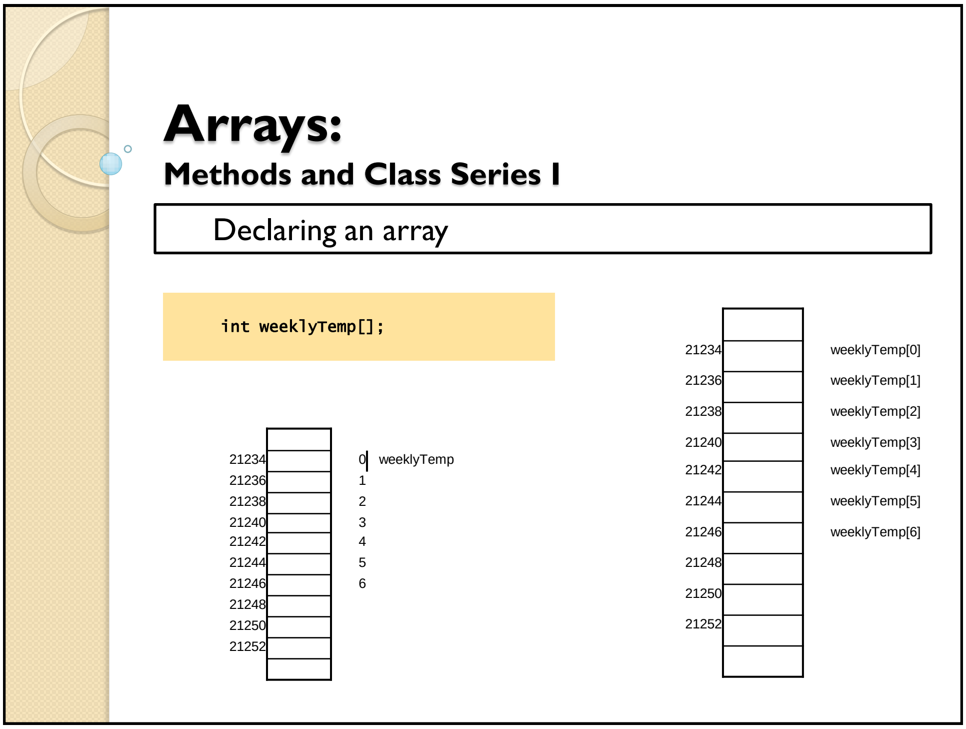
Arrays:

Methods and Class Series I

Array example:

```
int sun_tmp=55, mon_tmp=54, tue_tmp=56;
int wed_tmp=52, thu_tmp=51, fri_tmp=53, sat_tmp=50;

float av_tmp =
    (sun_tmp + mon_tmp + tue_tmp + wed_tmp +
            thu_tmp + fri_tmp + sat_tmp)/7;

System.out.printf("The Average Temperature is: %f", av_tmp


int tmp[]={55, 54, 56, 52, 51, 53, 50};

float av_tmp =
    (tmp[0] + tmp[1] + tmp[2] + tmp[3] +
            tmp[4] + tmp[5] + tmp[6])/7;

System.out.printf("The Average Temperature is: %f", av_tmp);
```

21234 weeklyTemp[0]
21236 weeklyTemp[1]
21238 weeklyTemp[2]
21240 weeklyTemp[3]
21242 weeklyTemp[4]
21244 weeklyTemp[5]
21246 weeklyTemp[6]
21248
21250
21252



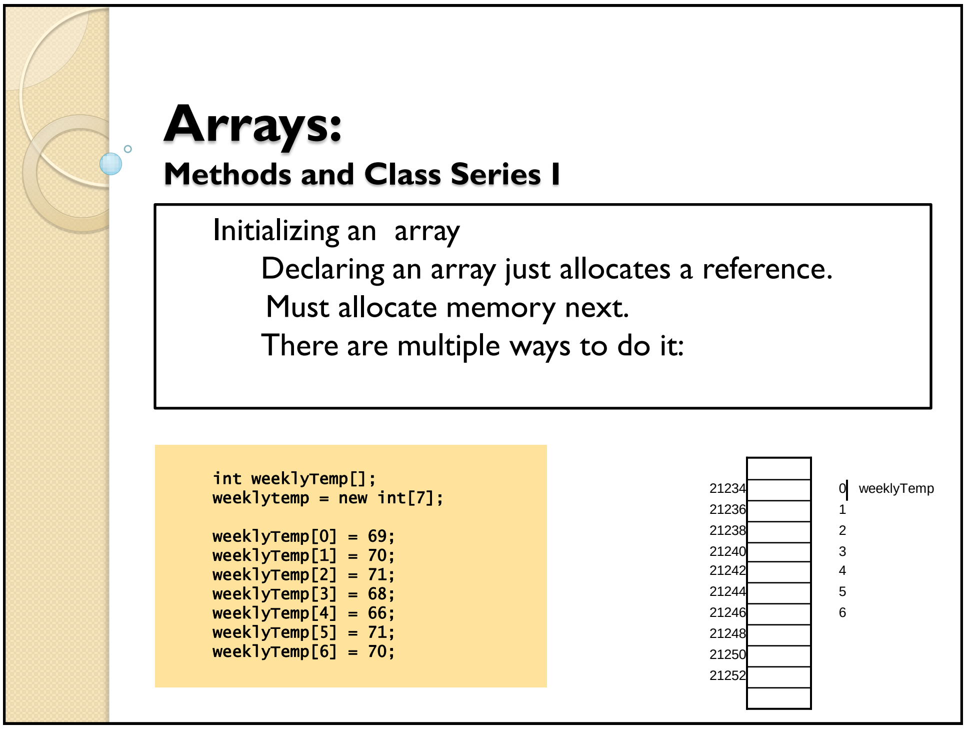
Arrays:

Methods and Class Series I

Declaring an array

```
int weeklyTemp[];
```

21234 0 weeklyTemp
21236 1
21238 2
21240 3
21242 4
21244 5
21246 6
21248
21250
21252
21234 weeklyTemp[0]
21236 weeklyTemp[1]
21238 weeklyTemp[2]
21240 weeklyTemp[3]
21242 weeklyTemp[4]
21244 weeklyTemp[5]
21246 weeklyTemp[6]
21248
21250
21252



Arrays:

Methods and Class Series I

Initializing an array

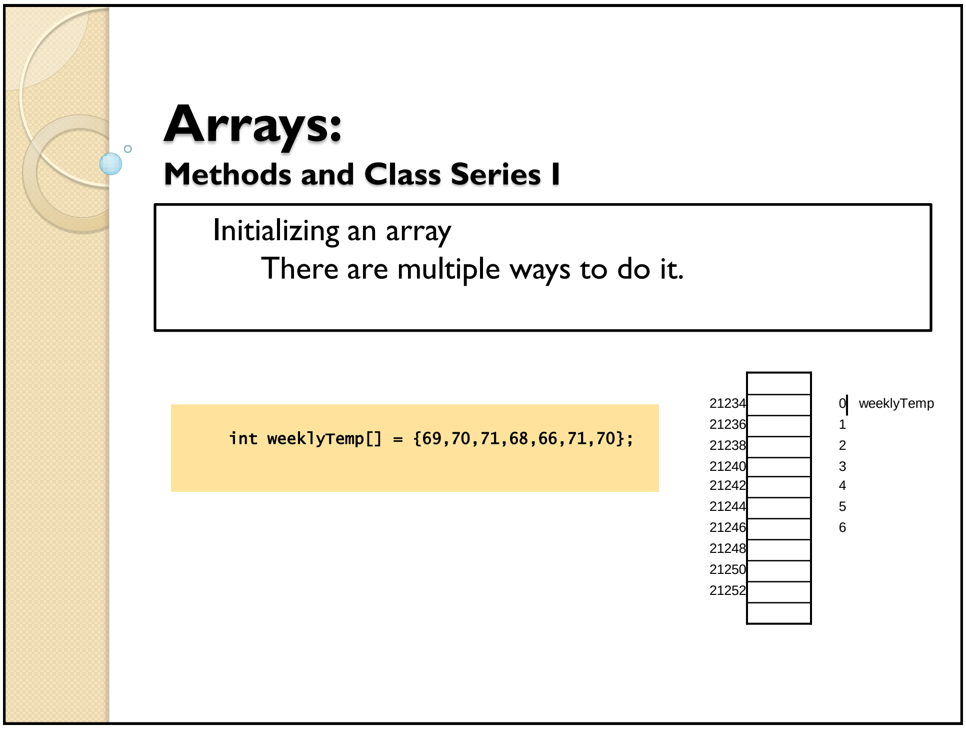
Declaring an array just allocates a reference.

Must allocate memory next.

There are multiple ways to do it:

```
int weeklyTemp[];
weeklytemp = new int[7];

weeklyTemp[0] = 69;
weeklyTemp[1] = 70;
weeklyTemp[2] = 71;
weeklyTemp[3] = 68;
weeklyTemp[4] = 66;
weeklyTemp[5] = 71;
weeklyTemp[6] = 70;
```

21234
21236
21238
21240
21242
21244
21246
21248
21250
21252
0
1
2
3
4
5
6
weeklyTemp



Arrays:

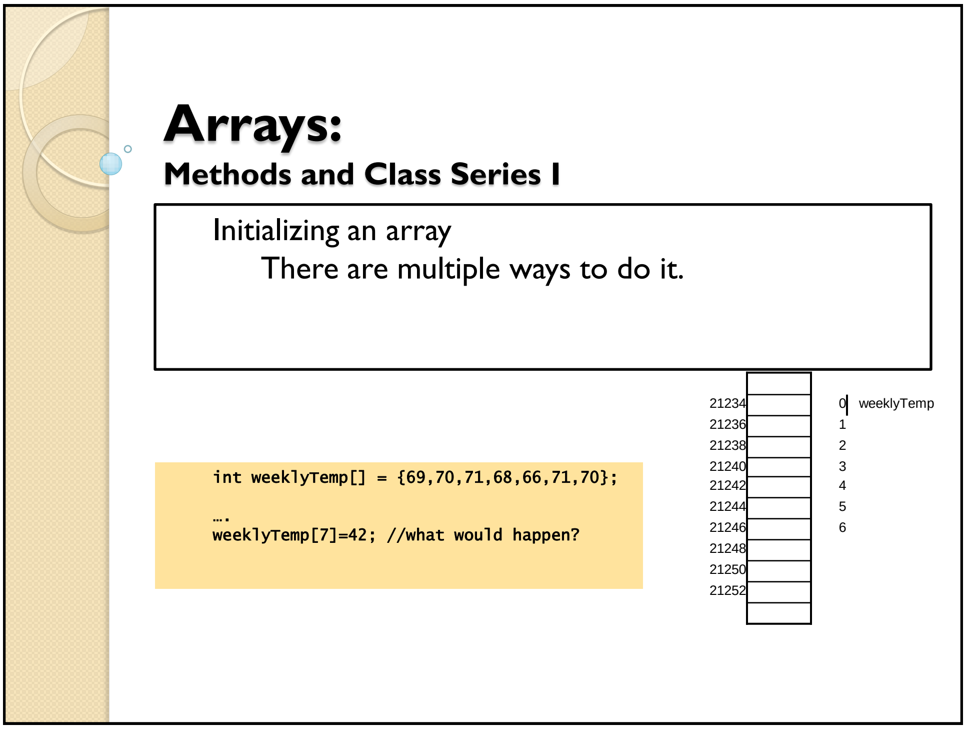
Methods and Class Series I

Initializing an array

There are multiple ways to do it.

```
int weeklyTemp[] = {69,70,71,68,66,71,70};
```

21234
21236
21238
21240
21242
21244
21246
21248
21250
21252
0
1
2
3
4
5
6
weeklyTemp



Arrays:

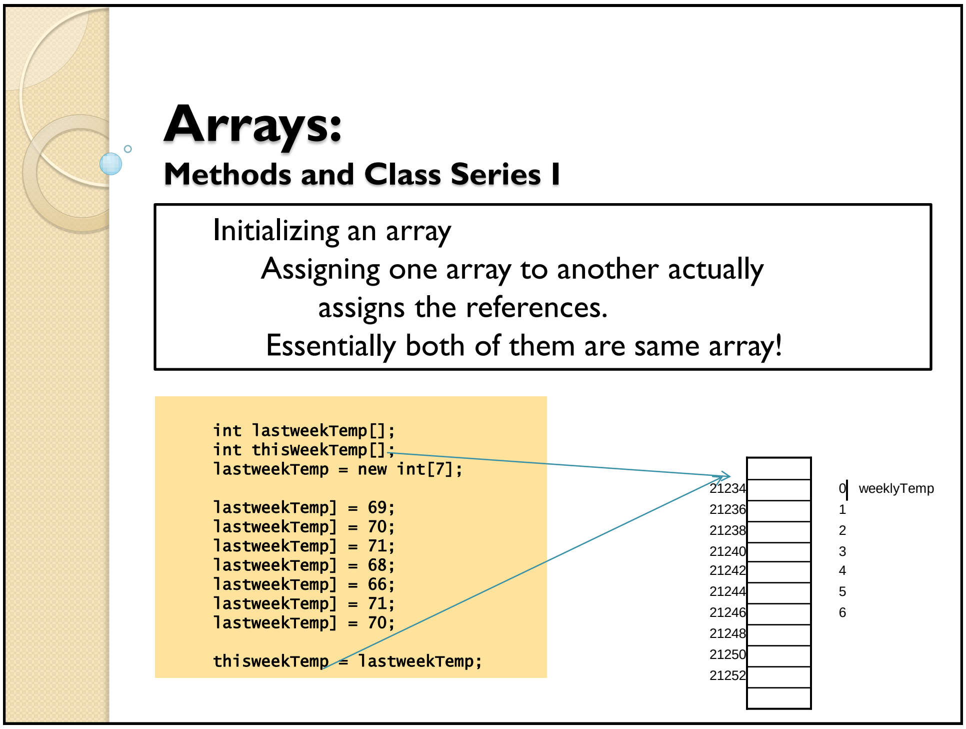
Methods and Class Series I

Initializing an array

There are multiple ways to do it.

```
int weeklyTemp[] = {69,70,71,68,66,71,70};

….
weeklyTemp[7]=42; //what would happen?
```

21234	0 weeklyTemp
21236	1
21238	2
21240	3
21242	4
21244	5
21246	6
21248	
21250	
21252	



Arrays:

Methods and Class Series I

Initializing an array

Assigning one array to another actually assigns the references.

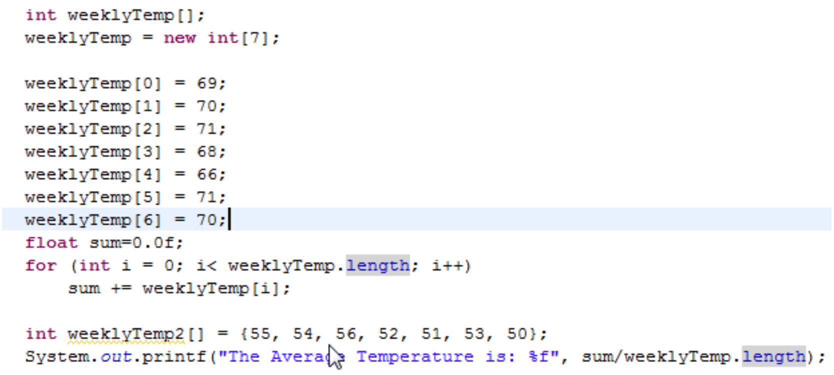
Essentially both of them are same array!

```
int lastweekTemp[];
int thisweekTemp[];
lastweekTemp = new int[7];

lastweekTemp] = 69;
lastweekTemp] = 70;
lastweekTemp] = 71;
lastweekTemp] = 68;
lastweekTemp] = 66;
lastweekTemp] = 71;
lastweekTemp] = 70;

thisweekTemp = lastweekTemp;
```

21234	0 weeklyTemp
21236	1
21238	2
21240	3
21242	4
21244	5
21246	6
21248	
21250	
21252	

Arrays:
Methods and Class Series I

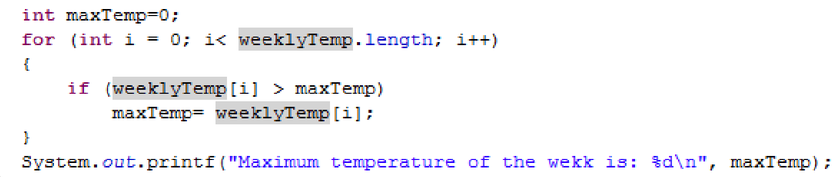
Usage of array
Easy for large sets of data.



```
int weeklyTemp[];
weeklyTemp = new int[7];

weeklyTemp[0] = 69;
weeklyTemp[1] = 70;
weeklyTemp[2] = 71;
weeklyTemp[3] = 68;
weeklyTemp[4] = 66;
weeklyTemp[5] = 71;
weeklyTemp[6] = 70;
float sum=0.0f;
for (int i = 0; i< weeklyTemp.length; i++)
    sum += weeklyTemp[i];

int weeklyTemp2[] = {55, 54, 56, 52, 51, 53, 50};
System.out.printf("The Average Temperature is: %f", sum/weeklyTemp.length);
```

Arrays:
Methods and Class Series I

Usage of array
Used in loop



```
int maxTemp=0;
for (int i = 0; i< weeklyTemp.length; i++)
{
    if (weeklyTemp[i] > maxTemp)
        maxTemp= weeklyTemp[i];
}
System.out.printf("Maximum temperature of the wekk is: %d\n", maxTemp);
```

Arrays:

Methods and Class Series I

weeklyTemp3

34500	55	0
34502	56	1
34504	67	2
34506	54	3
34508	67	4
34510	34	5
34512	34	6

Comparing arrays

Array name is reference.
When you compare values (other than built in type) you are comparing the reference.
Create own comparison if you need to.

weeklyTemp4

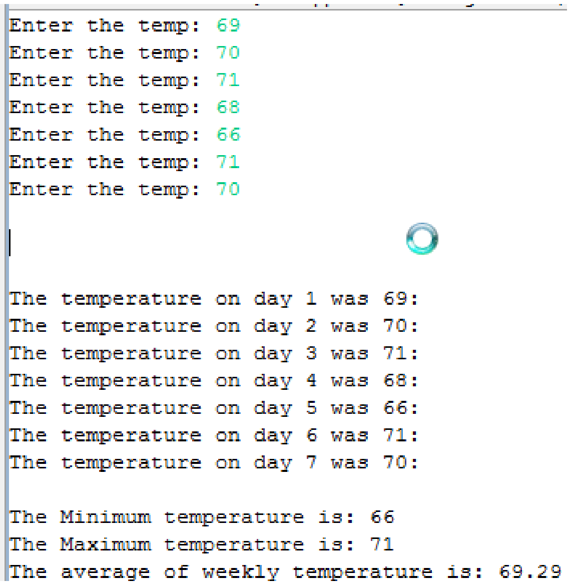
4522	55	0
4524	56	1
4526	67	2
4528	54	3
4530	67	4
4532	34	5
4534	34	6

```
int weeklyTemp3[] = {55, 56, 67, 54, 67, 34, 34};
int weeklyTemp4[] = {55, 56, 67, 54, 67, 34, 34};

if (weeklyTemp3 == weeklyTemp4)
    System.out.printf("These two are equal array");
else
    System.out.printf("These two are un-equal array");
```

Arrays

Methods an

- Demo



```
Enter the temp: 69
Enter the temp: 70
Enter the temp: 71
Enter the temp: 68
Enter the temp: 66
Enter the temp: 71
Enter the temp: 70


The temperature on day 1 was 69:
The temperature on day 2 was 70:
The temperature on day 3 was 71:
The temperature on day 4 was 68:
The temperature on day 5 was 66:
The temperature on day 6 was 71:
The temperature on day 7 was 70:

The Minimum temperature is: 66
The Maximum temperature is: 71
The average of weekly temperature is: 69.29
```

Array example:

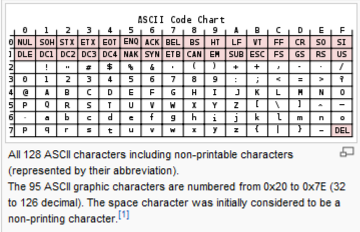
Character Strings:

Methods and Class Series II

Problem at hand that needs a solution

Character Strings:

Methods and Class Series II



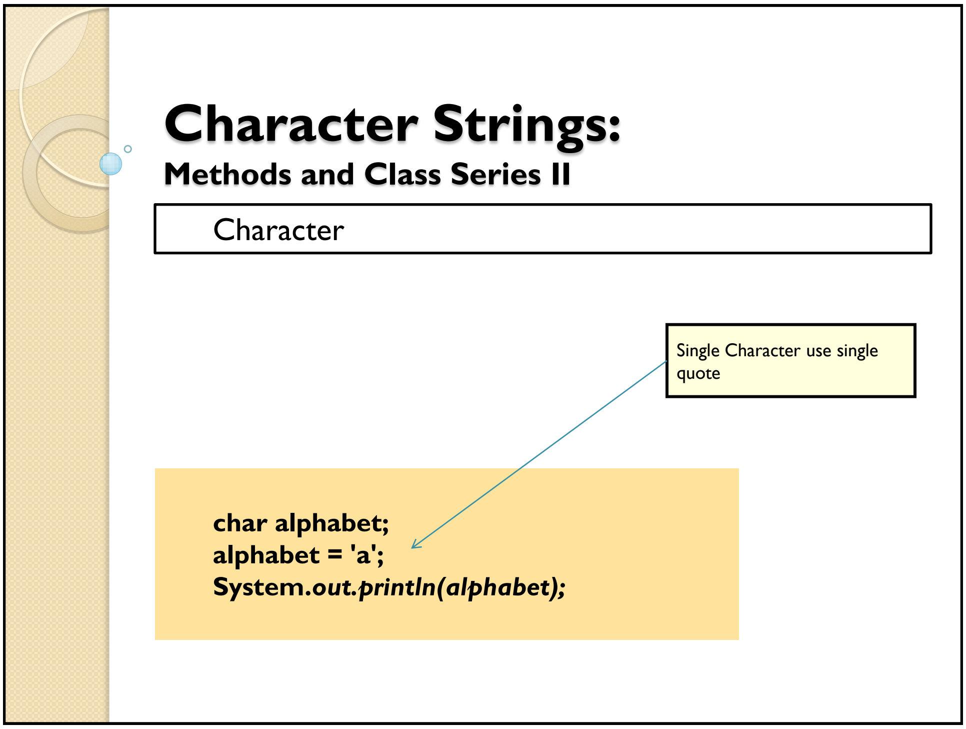
ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

All 128 ASCII characters including non-printable characters (represented by their abbreviation).
The 95 ASCII graphic characters are numbered from 0x20 to 0x7E (32 to 126 decimal). The space character was initially considered to be a non-printing character.[1]

Characters in Java
- Printable and nonprintable
- Lowercase letters
- Uppercase letters
- Numbers
- Special characters

(refer to the chart above from Wiki)



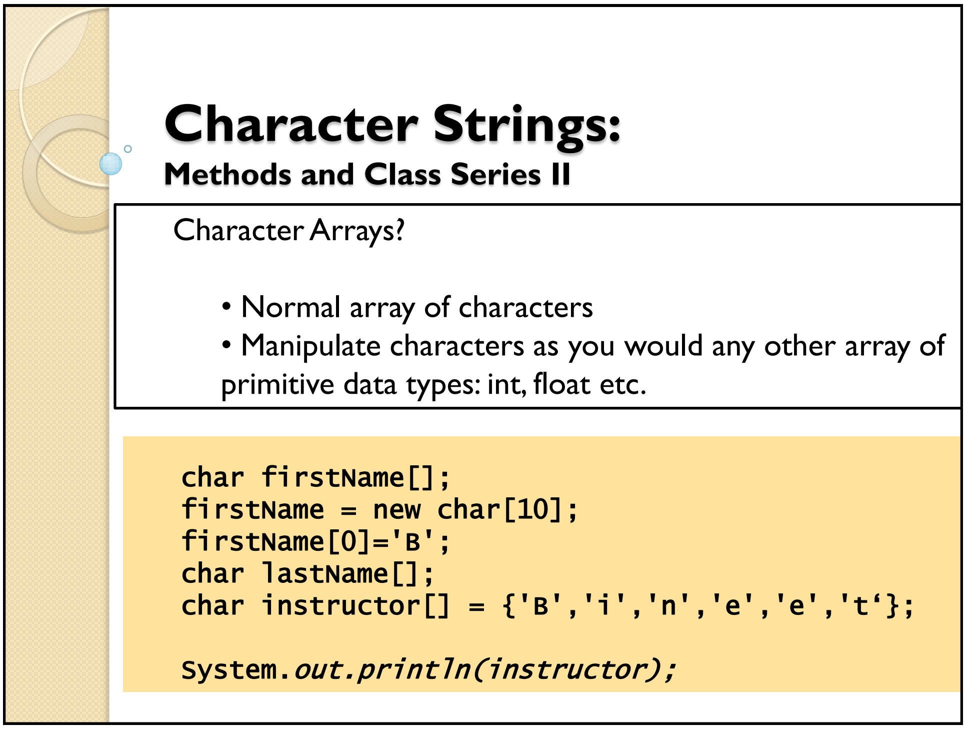
Character Strings:

Methods and Class Series II

Character

Single Character use single quote

```
char alphabet;
alphabet = 'a';
System.out.println(alphabet);
```



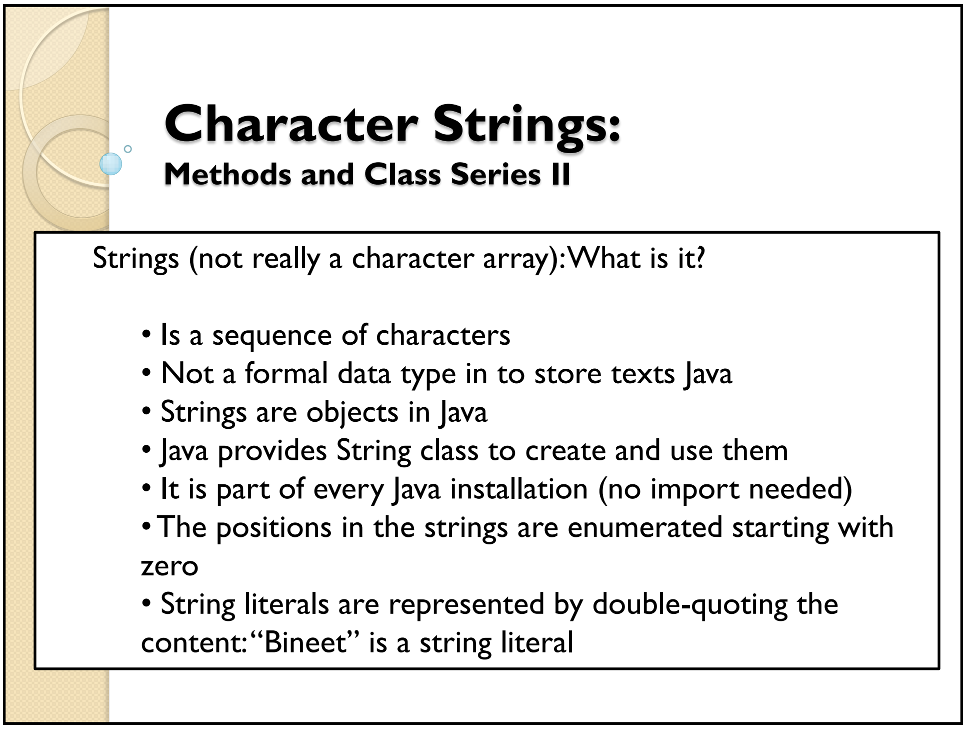
Character Strings:

Methods and Class Series II

Character Arrays?

- Normal array of characters
- Manipulate characters as you would any other array of primitive data types: int, float etc.

```
char firstName[];
firstName = new char[10];
firstName[0]='B';
char lastName[];
char instructor[] = {'B','i','n','e','e','t‘};

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

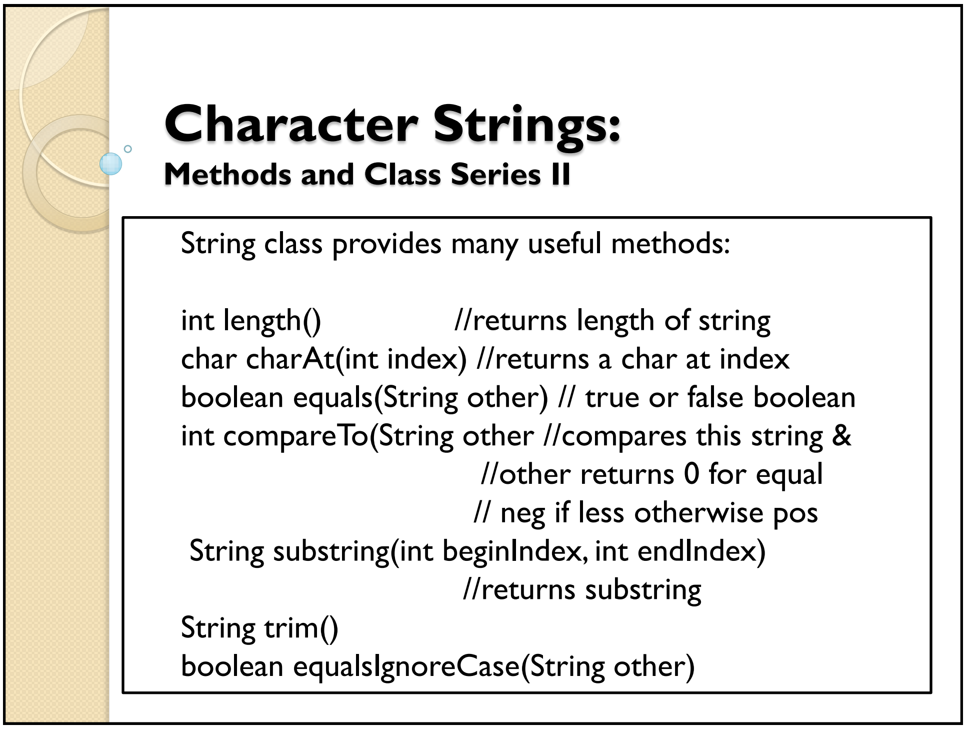


Character Strings:

Methods and Class Series II

Strings (not really a character array): What is it?

- Is a sequence of characters
- Not a formal data type in to store texts Java
- Strings are objects in Java
- Java provides String class to create and use them
- It is part of every Java installation (no import needed)
- The positions in the strings are enumerated starting with zero
- String literals are represented by double-quoting the content: "Bineet" is a string literal

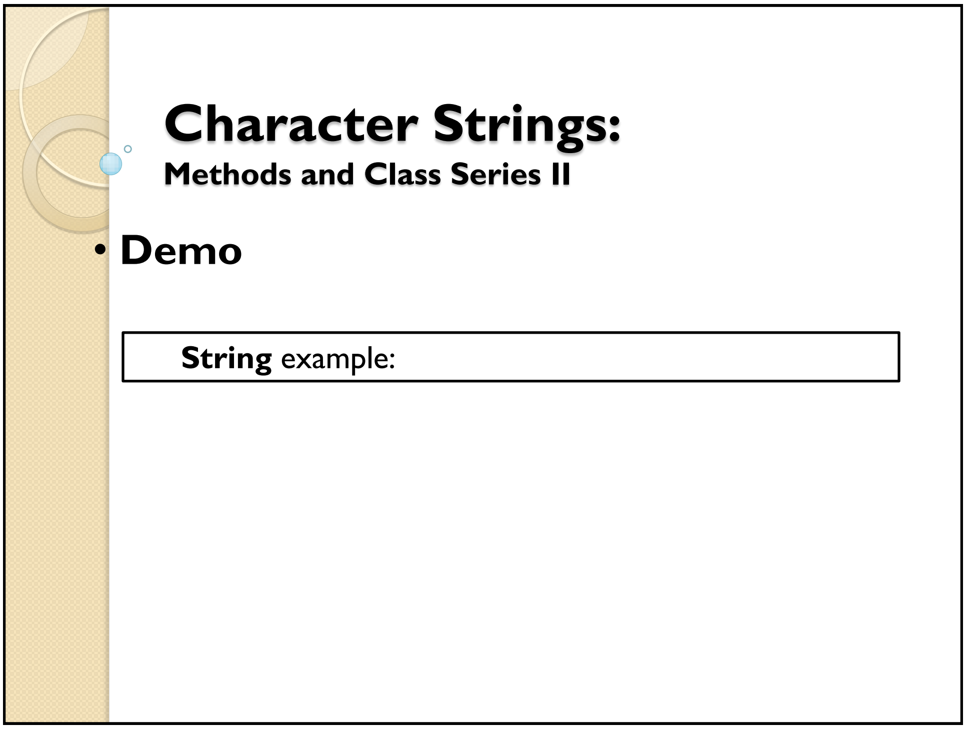


Character Strings:

Methods and Class Series II

String class provides many useful methods:

```
int length()              //returns length of string
char charAt(int index) //returns a char at index
boolean equals(String other) // true or false boolean
int compareTo(String other //compares this string &
                              //other returns 0 for equal
                             // neg if less otherwise pos
 String substring(int beginIndex, int endIndex)
                             //returns substring
String trim()
boolean equalsIgnoreCase(String other)
```

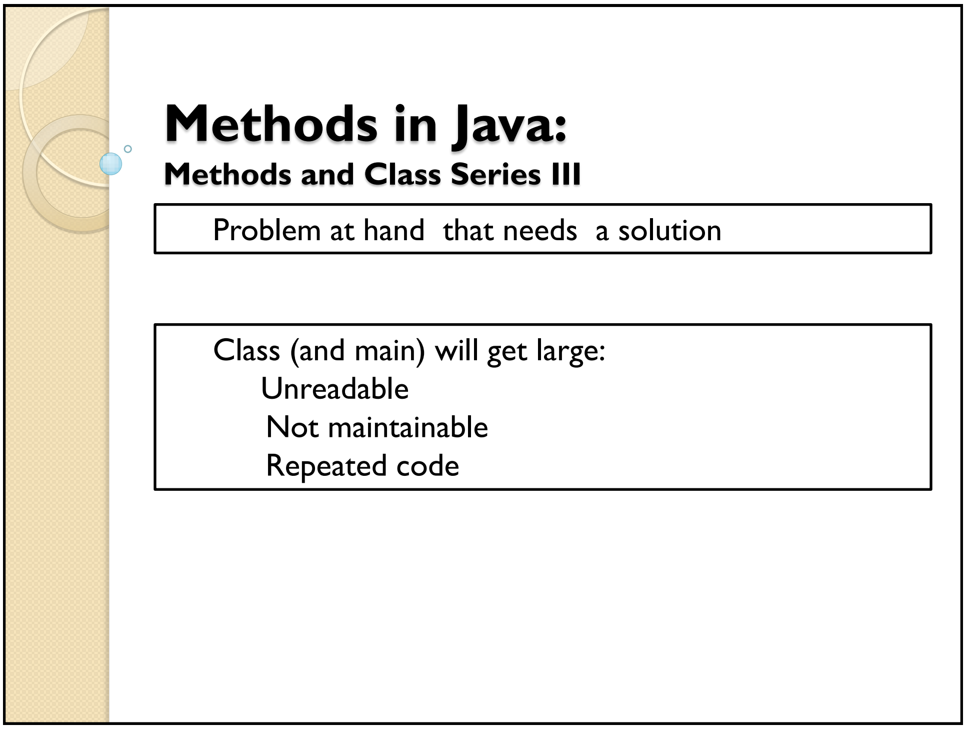


Character Strings:

Methods and Class Series II

- Demo

String example:

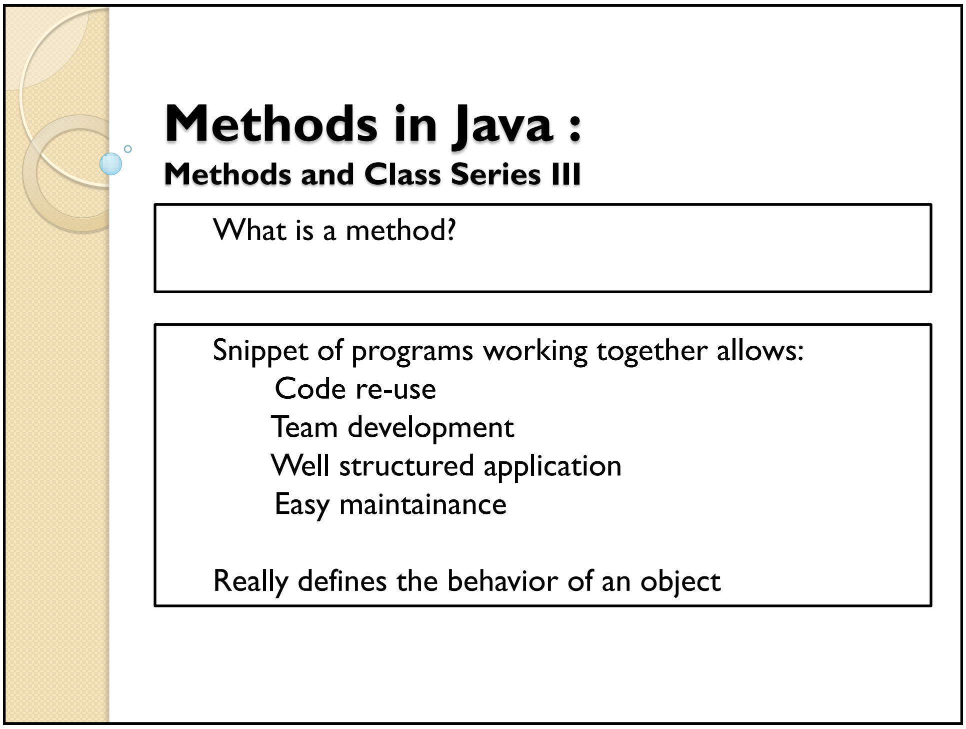


Methods in Java:

Methods and Class Series III

Problem at hand that needs a solution

Class (and main) will get large:
Unreadable
Not maintainable
Repeated code



Methods in Java :

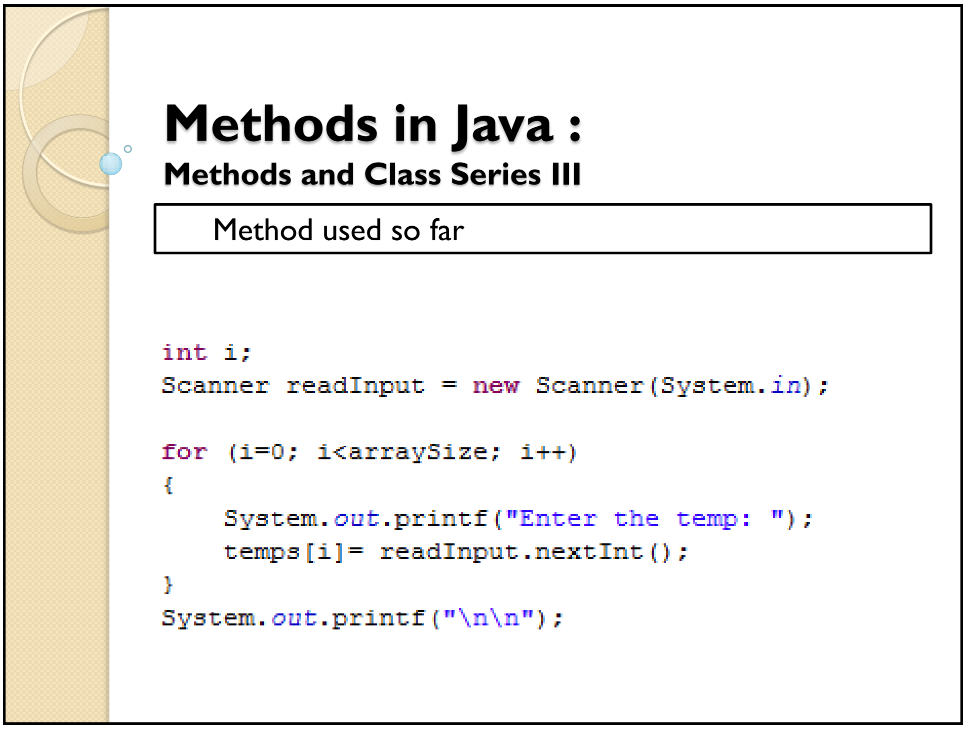
Methods and Class Series III

What is a method?

Snippet of programs working together allows:

- Code re-use
- Team development
- Well structured application
- Easy maintainance

Really defines the behavior of an object

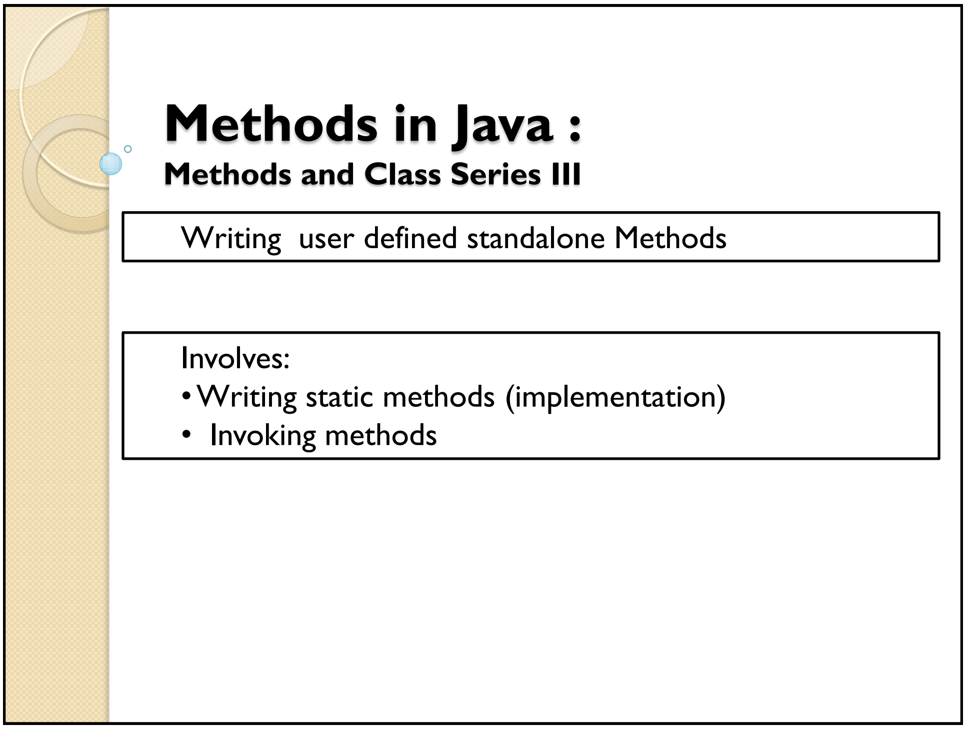


Methods in Java :

Methods and Class Series III

Method used so far

```
int i;
Scanner readInput = new Scanner(System.in);

for (i=0; i<arraySize; i++)
{
     System.out.printf("Enter the temp: ");
     temps[i]= readInput.nextInt();
}
System.out.printf("\n\n");
```



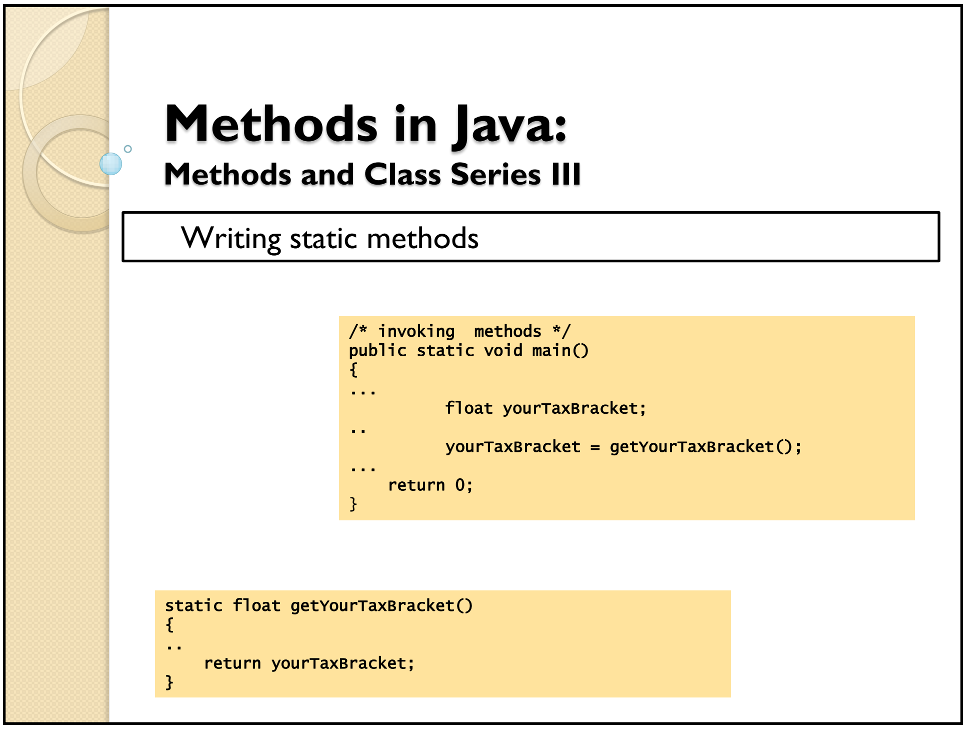
Methods in Java :

Methods and Class Series III

Writing user defined standalone Methods

Involves:

- Writing static methods (implementation)
- Invoking methods



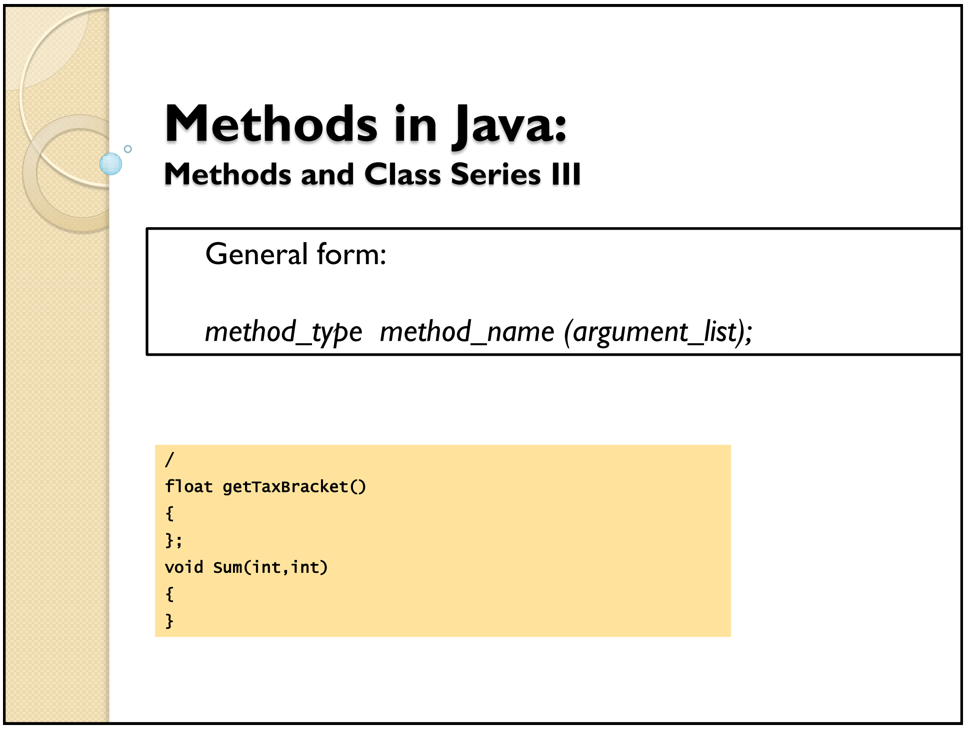
Methods in Java:

Methods and Class Series III

Writing static methods

```
/* invoking  methods */
public static void main()
{
...
          float yourTaxBracket;
..
          yourTaxBracket = getYourTaxBracket();
...
    return 0;
}
```

```
static float getYourTaxBracket()
{
..
    return yourTaxBracket;
}
```



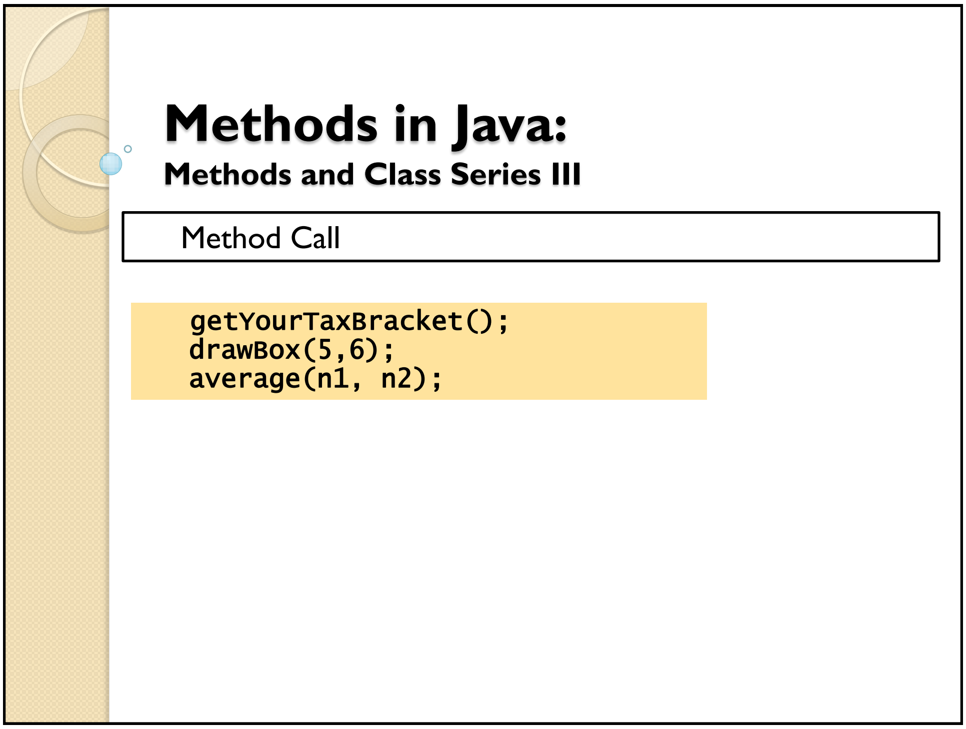
Methods in Java:

Methods and Class Series III

General form:

method_type method_name (argument_list);

```
/
float getTaxBracket()
{
};
void Sum(int,int)
{
}
```



Methods in Java:

Methods and Class Series III

Method Call

```
   getYourTaxBracket();
   drawBox(5,6);
   average(n1, n2);
```

Methods in Java:

Methods and Class Series III

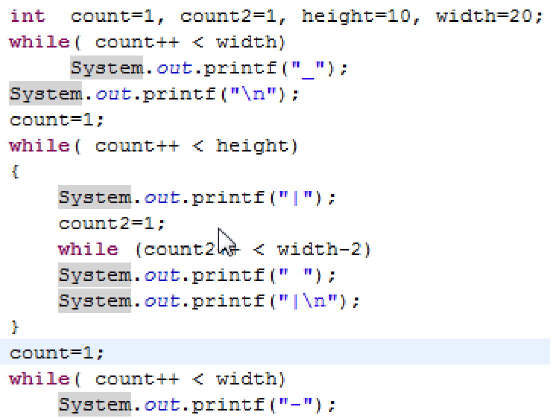
Method Definition

```
/* method definition */
static float getYourTaxBracket()
{
..
    return yourTaxBracket;
}
```

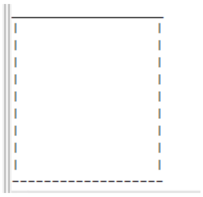
Methods in Java:

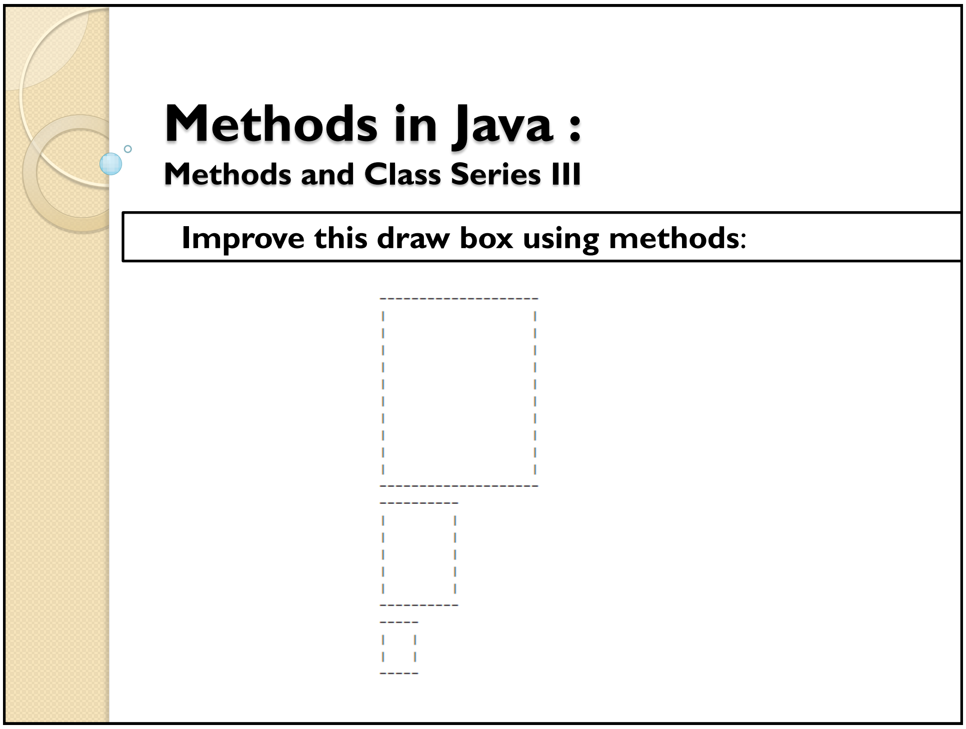
Methods and Class Series III

Improve this draw box using methods:



```
int  count=1, count2=1, height=10, width=20;
while( count++ < width)
     System.out.printf("_");
System.out.printf("\n");
count=1;
while( count++ < height)
{
    System.out.printf("|");
    count2=1;
    while (count2++ < width-2)
    System.out.printf(" ");
    System.out.printf("|\n");
}
count=1;
while( count++ < width)
    System.out.printf("-");
```



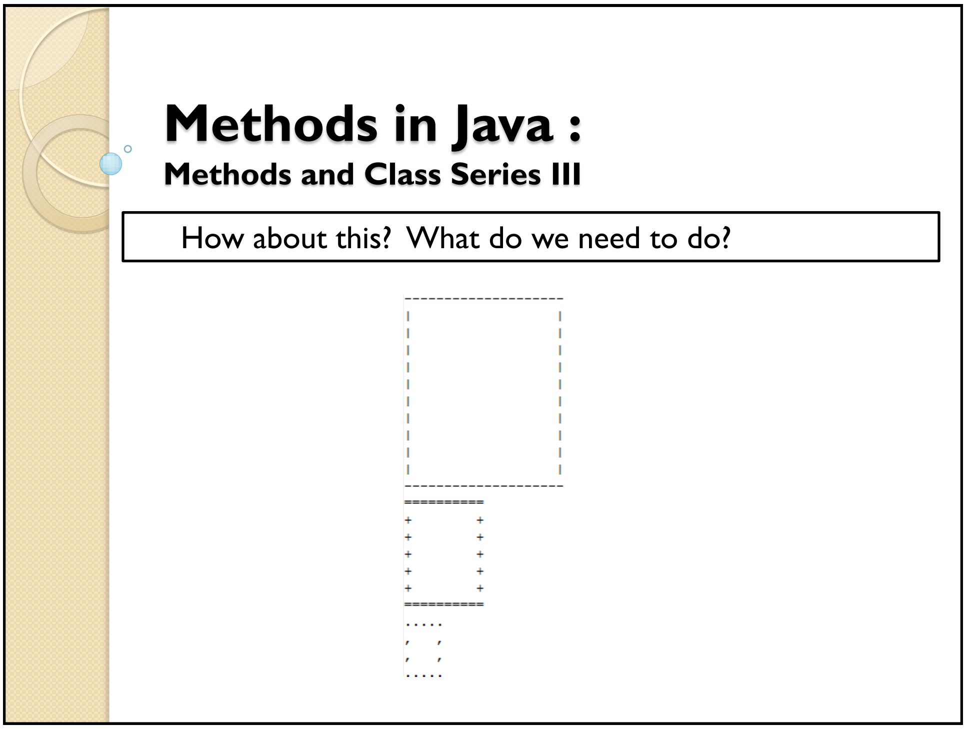


Methods in Java :

Methods and Class Series III

Improve this draw box using methods:

```
--------------------
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
--------------------
----------
|        |
|        |
|        |
|        |
|        |
----------
-----
|   |
|   |
-----
```



Methods in Java :

Methods and Class Series III

How about this? What do we need to do?

```
--------------------
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
|                  |
--------------------
==========
+        +
+        +
+        +
+        +
+        +
==========
.....
,   ,
,   ,
.....
```

Methods in Java :

Methods and Class Series III

- Demo

Method example:

Class and Objects:

Methods and Class Series IV

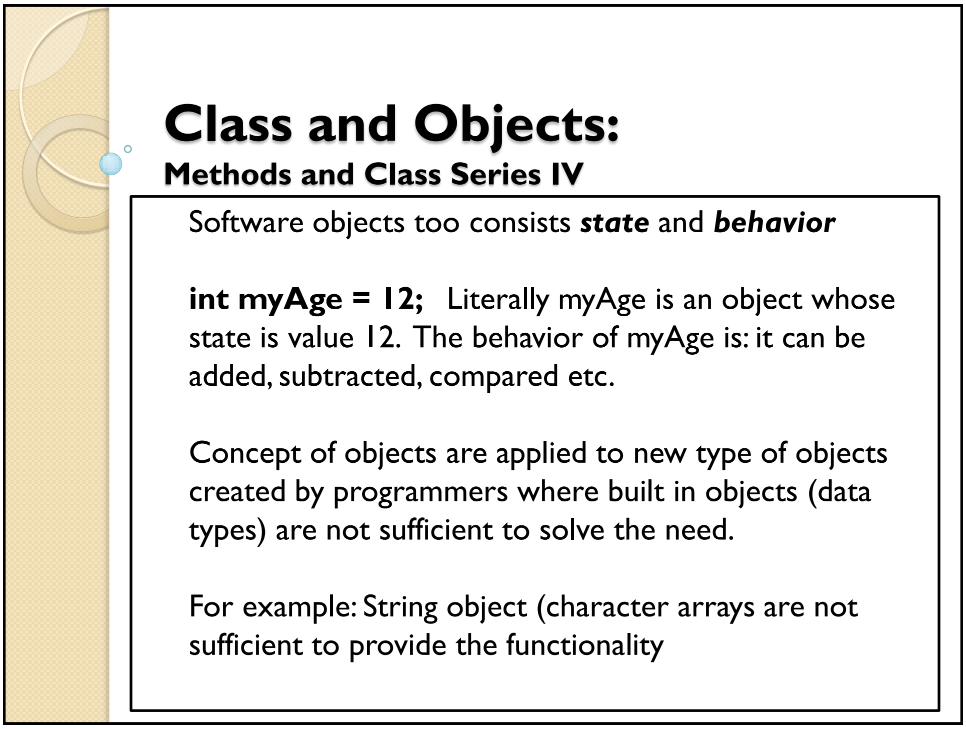
Java is a object oriented programming
Essentials to learn about objects

Real-world objects: computer, desk, dog, car

All real-world object has ***state*** and ***behavior***

A dog's state are his/her color, breed, name
A dog's behavior are barking, fetching, wagging

Identifying state and behavior of real-world object leads to thinking in OO programming



Class and Objects:

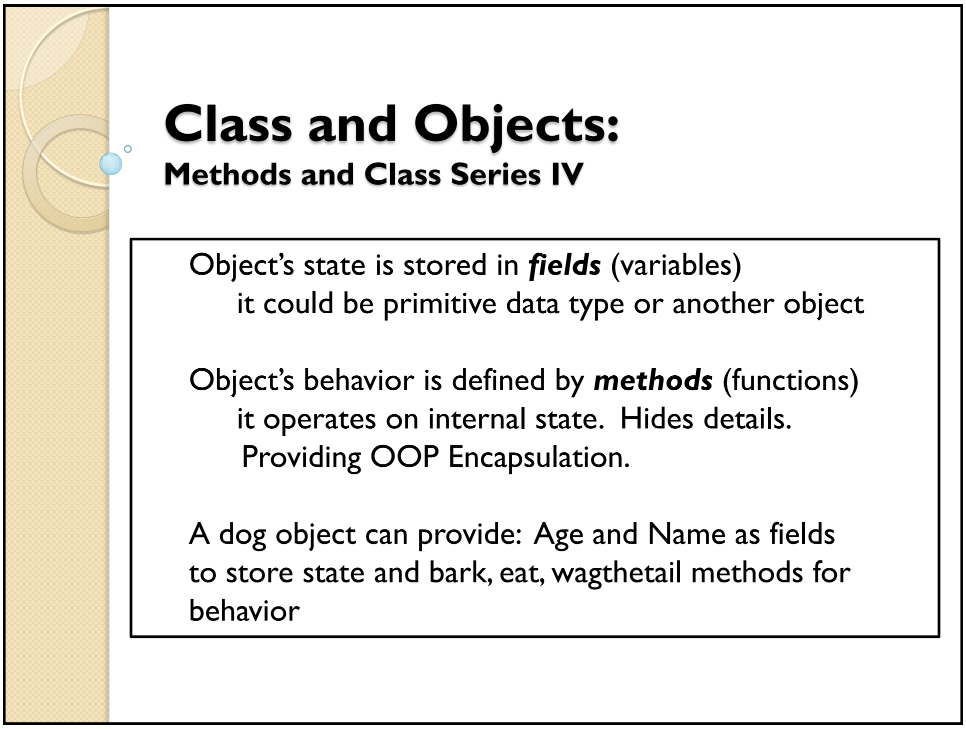
Methods and Class Series IV

Software objects too consists ***state*** and ***behavior***

int myAge = 12; Literally myAge is an object whose state is value 12. The behavior of myAge is: it can be added, subtracted, compared etc.

Concept of objects are applied to new type of objects created by programmers where built in objects (data types) are not sufficient to solve the need.

For example: String object (character arrays are not sufficient to provide the functionality



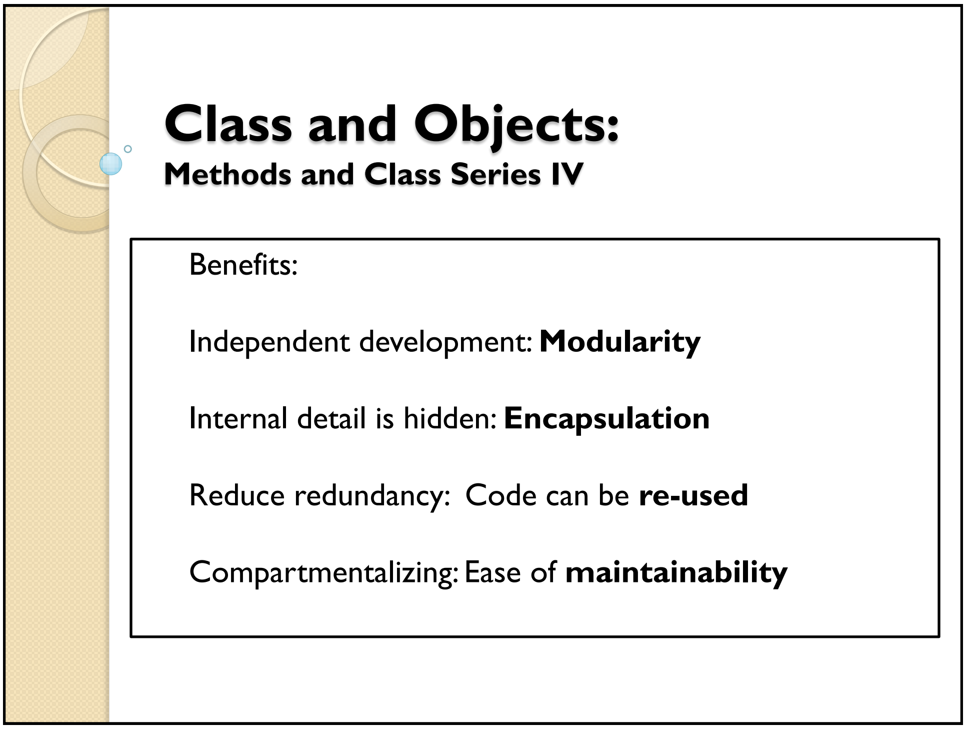
Class and Objects:

Methods and Class Series IV

Object's state is stored in ***fields*** (variables)
it could be primitive data type or another object

Object's behavior is defined by ***methods*** (functions)
it operates on internal state. Hides details.
Providing OOP Encapsulation.

A dog object can provide: Age and Name as fields to store state and bark, eat, wagthetail methods for behavior



Class and Objects:

Methods and Class Series IV

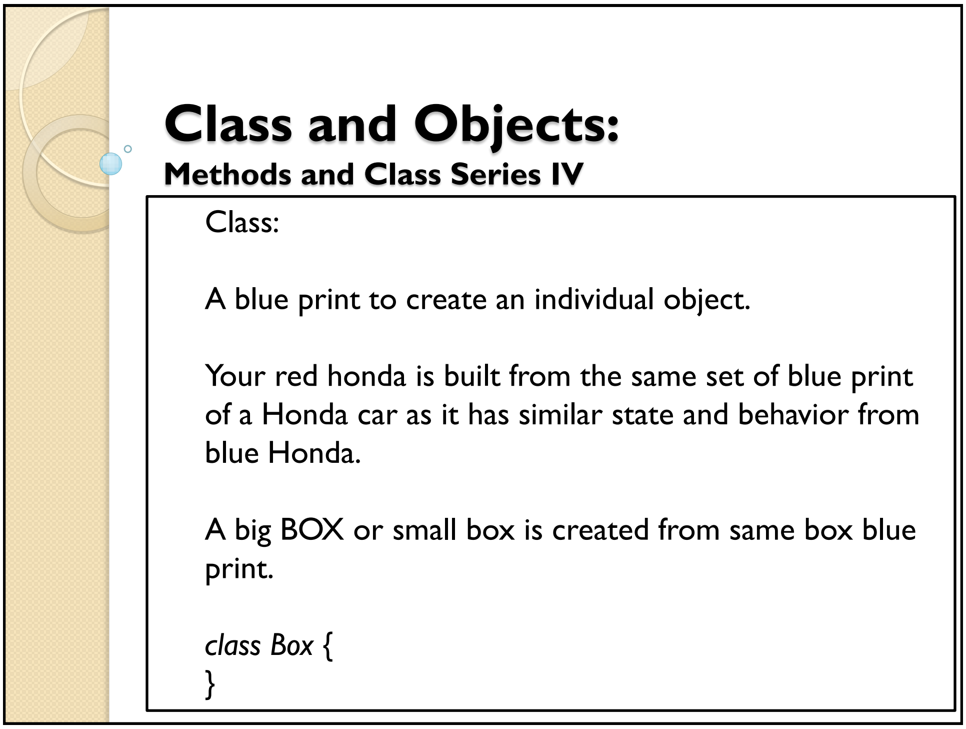
Benefits:

Independent development: **Modularity**

Internal detail is hidden: **Encapsulation**

Reduce redundancy: Code can be **re-used**

Compartmentalizing: Ease of **maintainability**



Class and Objects:

Methods and Class Series IV

Class:

A blue print to create an individual object.

Your red honda is built from the same set of blue print of a Honda car as it has similar state and behavior from blue Honda.

A big BOX or small box is created from same box blue print.

```
class Box {
}
```

Class and Objects:

Methods and Class Series IV

Object:

A red honda is an instance (object) of a generic Honda car (class)

Honda redHonda = new Honda();
Honda blueHonda = new Honda();

A big BOX or a small box is an instance (object) of a box blue print (class)

Box bigBox = new Box();
Box smallBox = new Box();

Class and Objects:

Methods and Class Series IV

Object state and behavior:

Box bigBox = new Box(20, 30);
Box smallBox = new Box(5, 10);

bigBox.drawYourself();
smallBox.drawYourself(();

bigBox.changeVLineSymbol("="");
bigBox.drawYourself();

Class and Objects:

Methods and Class Series IV

Inheritance:

Different kinds of objects have commonality and differences: Some additional features

Mountain bikes have gears, while mountain bikes also have two wheels like road bike.

Specialized objects *inherit* the commonality from other classes

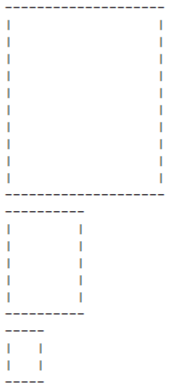
subclass inherits commonality from *superclasses*
class SolidBox ***extends*** *Box*{ …}

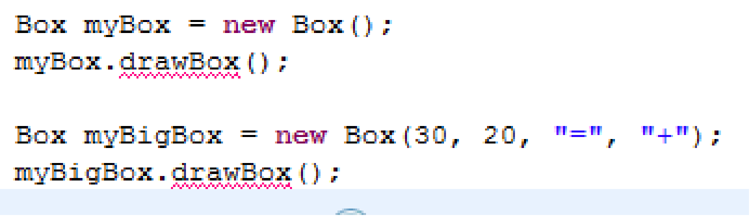
Class and Objects:

Methods and Class Series IV

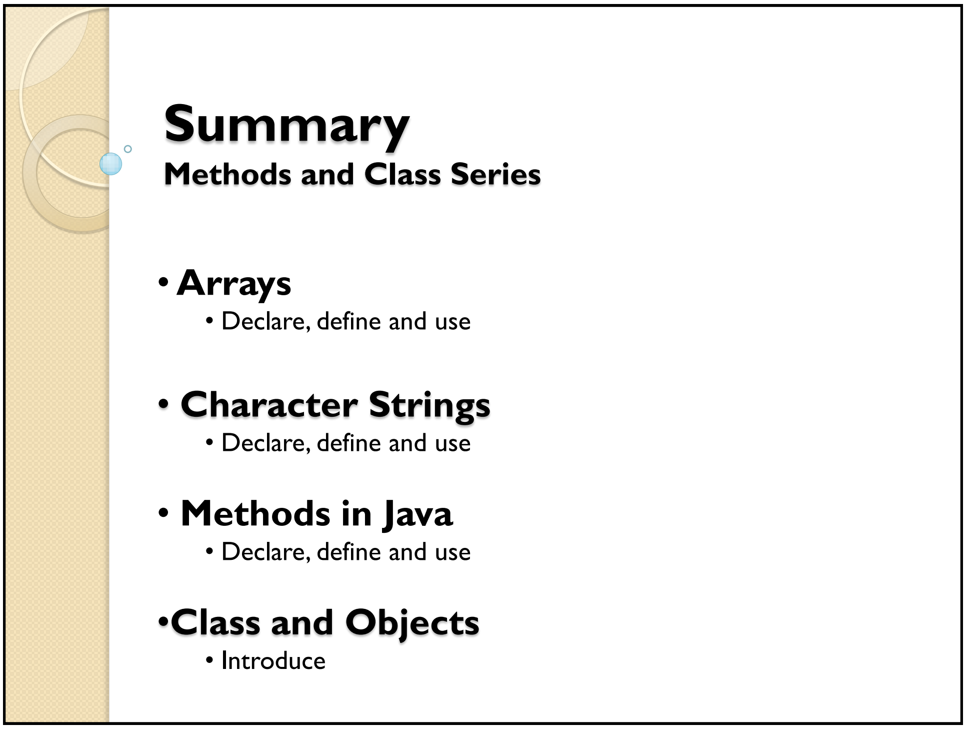
- **Demo**

Write your class:





```
Box myBox = new Box();
myBox.drawBox();

Box myBigBox = new Box(30, 20, "=", "+");
myBigBox.drawBox();
```



Summary

Methods and Class Series

- **Arrays**
  - Declare, define and use
- **Character Strings**
  - Declare, define and use
- **Methods in Java**
  - Declare, define and use
- **Class and Objects**
  - Introduce