ESc101 : Fundamental of Computing

I Semester 2008-09

Lecture 29

- Multi-dimensional arrays
- Characters and Srings

Multi-dimensional Arrays

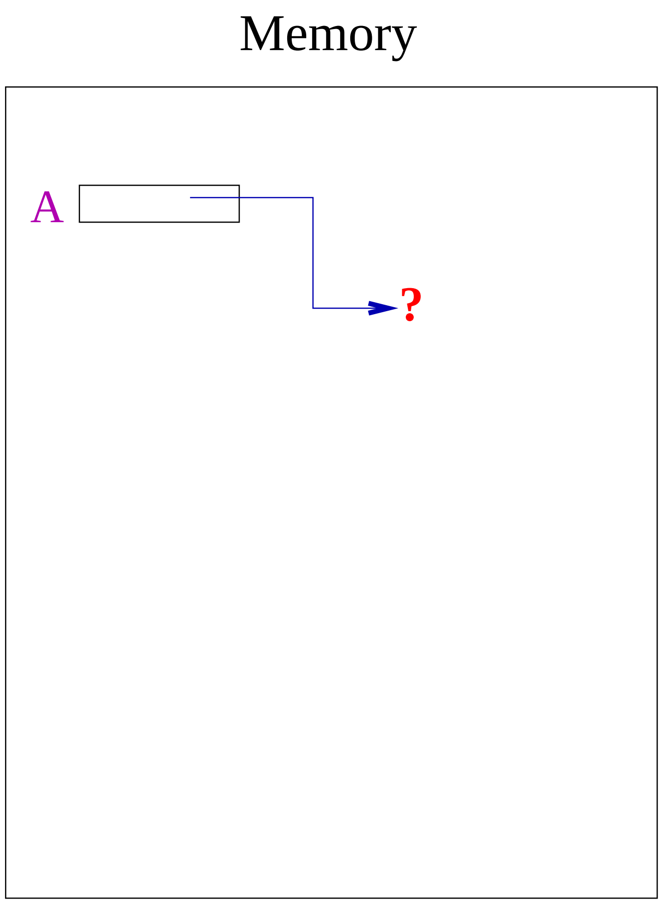
Recall the definition of array

Array : An Object which is an ordered collection of data items. These data items could be

- primitive types.
- references to objects of a class.

Array : declaration

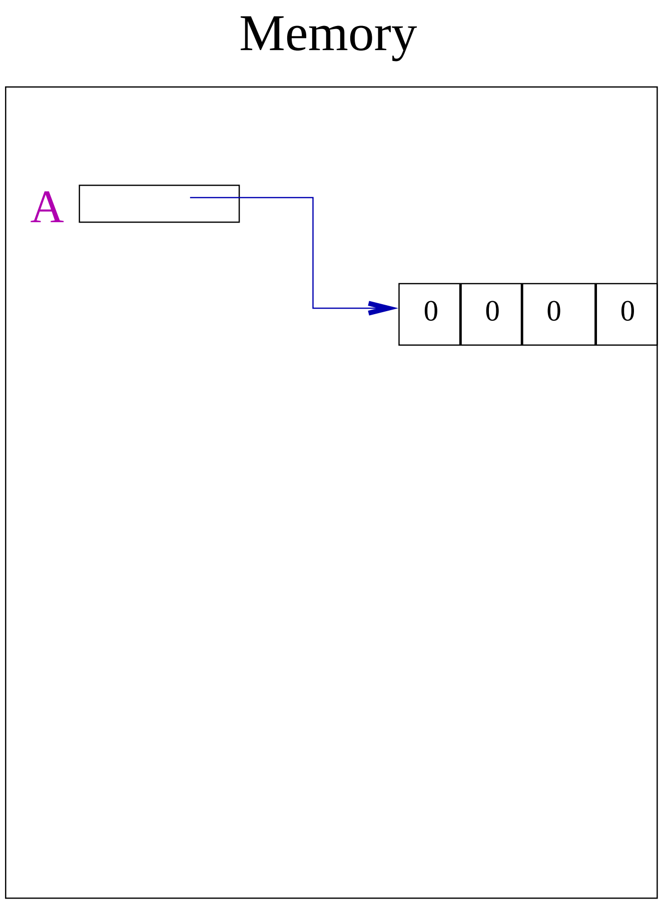
```
int[ ]    A  ;
```



Array : declaration and creation

```
int[ ]   A  ;

 A  = new int[4];
```

Expression of type int



What about array of “*Objects*”

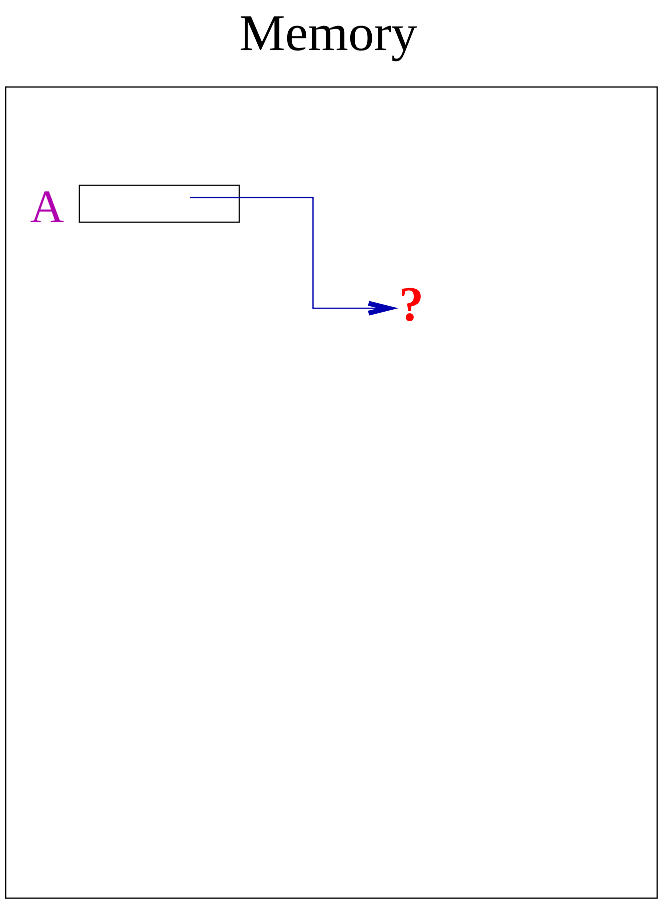
What about array of “*Objects*”

array whose data items are

references to objects of a class

Array of Points

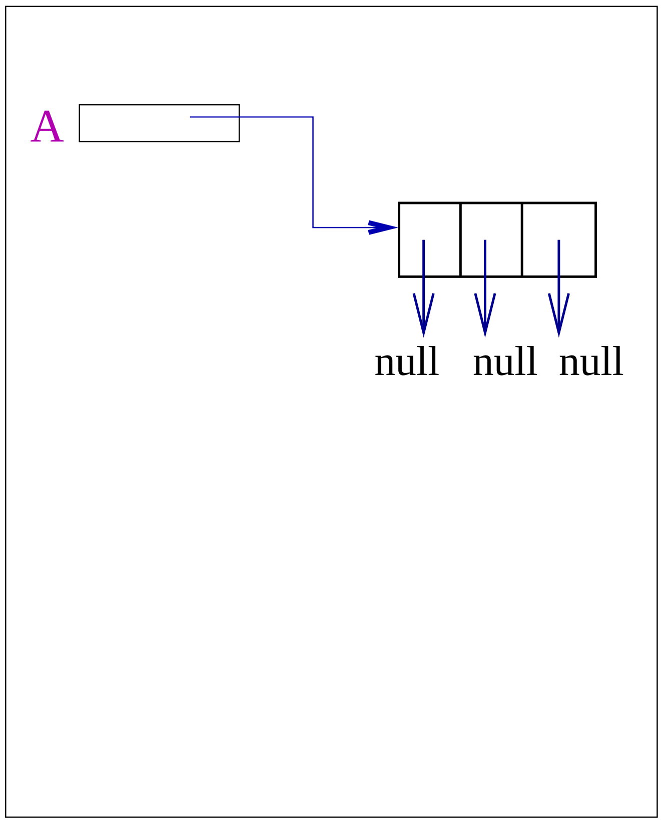
Point[ ] A ;



Array of Points

```
Point[ ] A ;

A  = new Point[3];
```

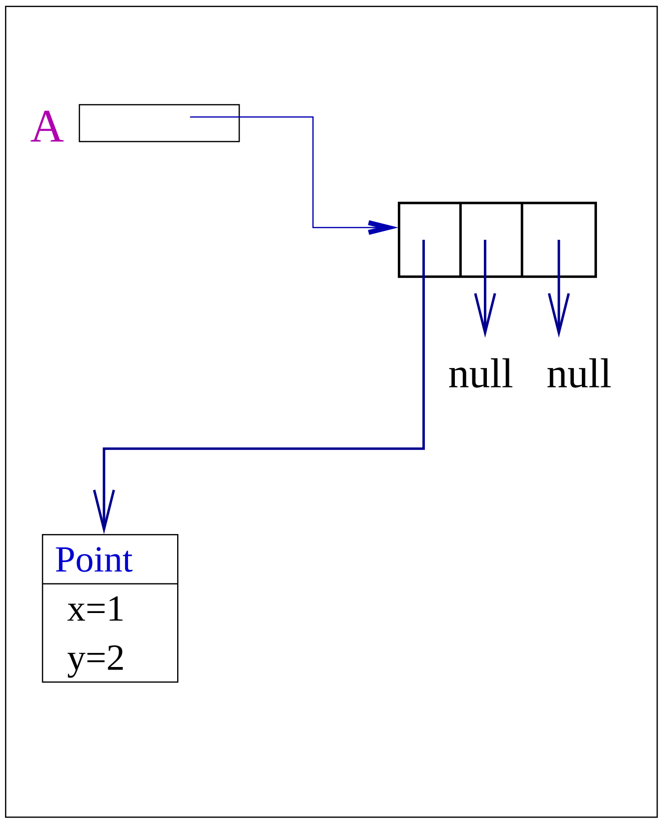
Memory



Array of Points

```
Point[ ] A ;

A  = new Point[3];

A[0] = new Point(1,2);
```

Memory



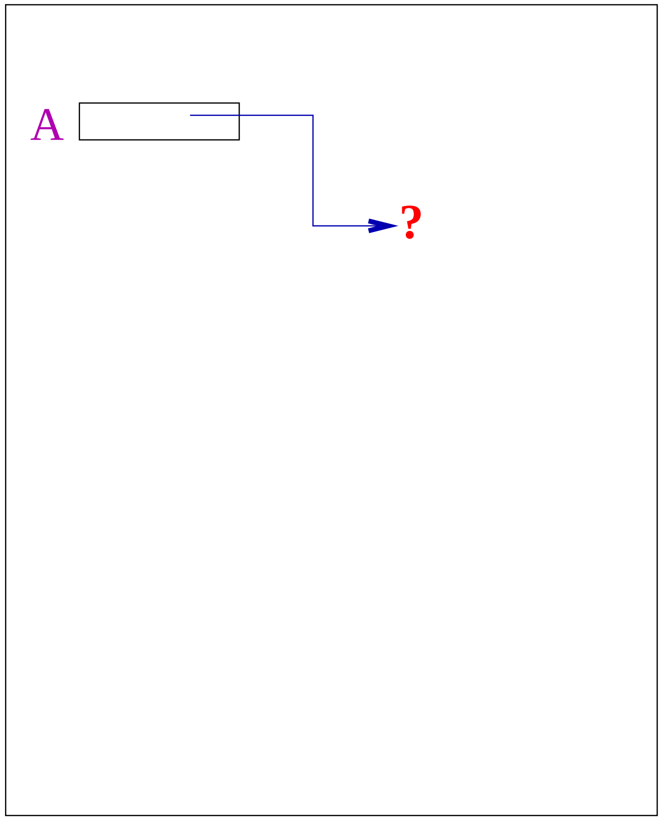
Array of arrays

The arrays whose data items are references to objects which are arrays.

Array of integer arrays : declaration

```
int[ ] [ ]    A  ;
```

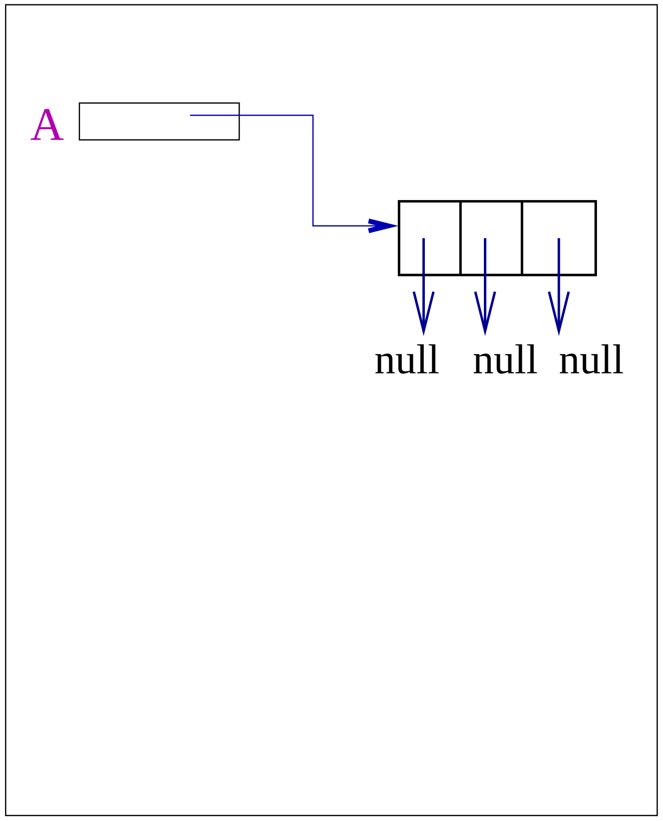
Memory



Array of integer arrays

```
int[ ][ ]  A ;

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

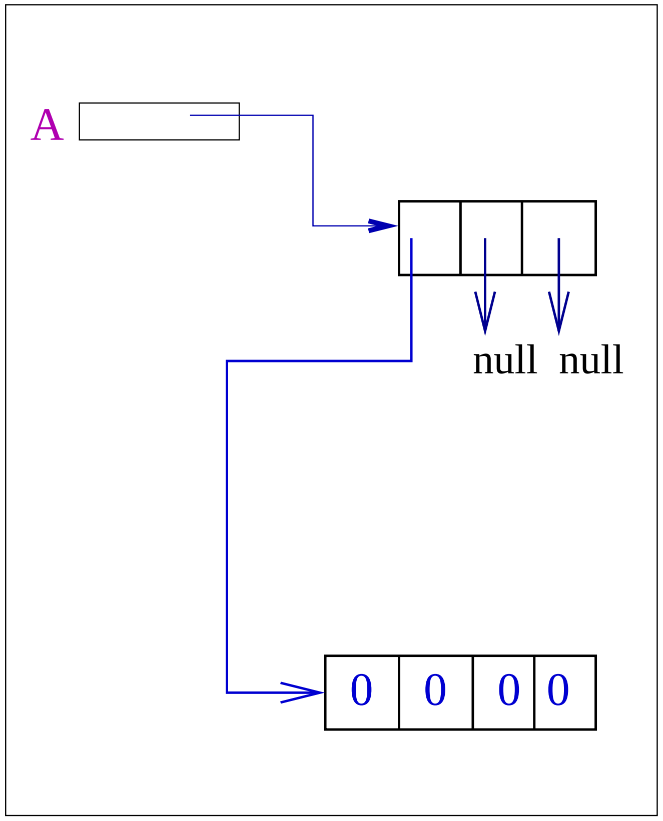
Memory



Array of integer arrays

```
int[ ][ ] A ;

A  = new int[3][ ];

A [0] = new int[4];
```

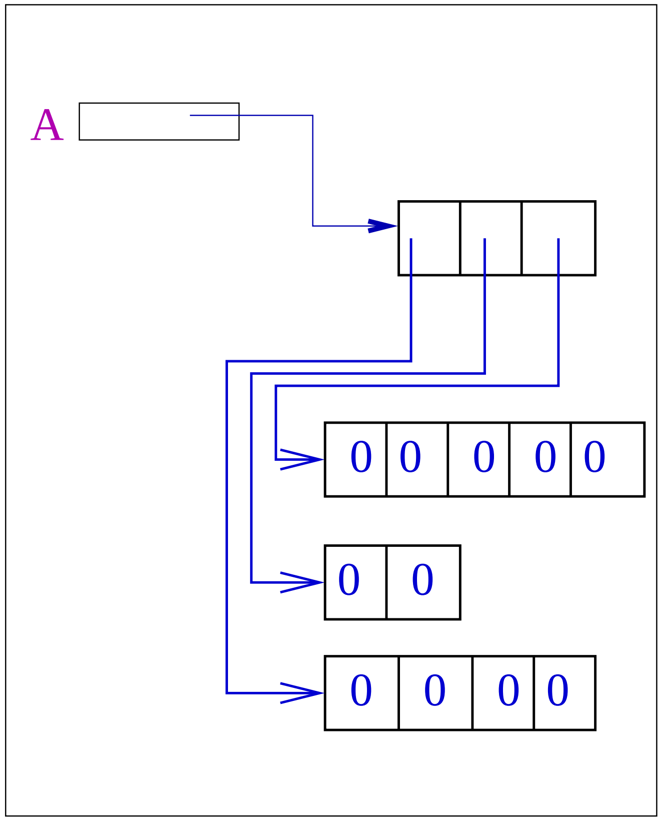
Memory



Array of integer arrays

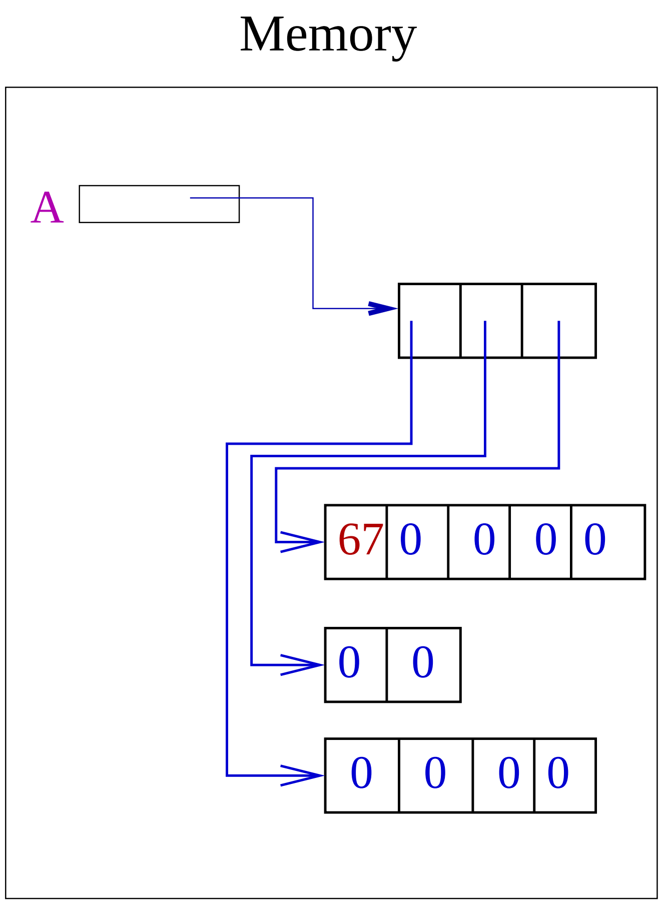
```
int[ ][ ] A ;
A  = new int[3][ ];
A [0] = new int[4];
A [1] = new int[2];
A [2] = new int[5];
```

Memory



Array of integer arrays

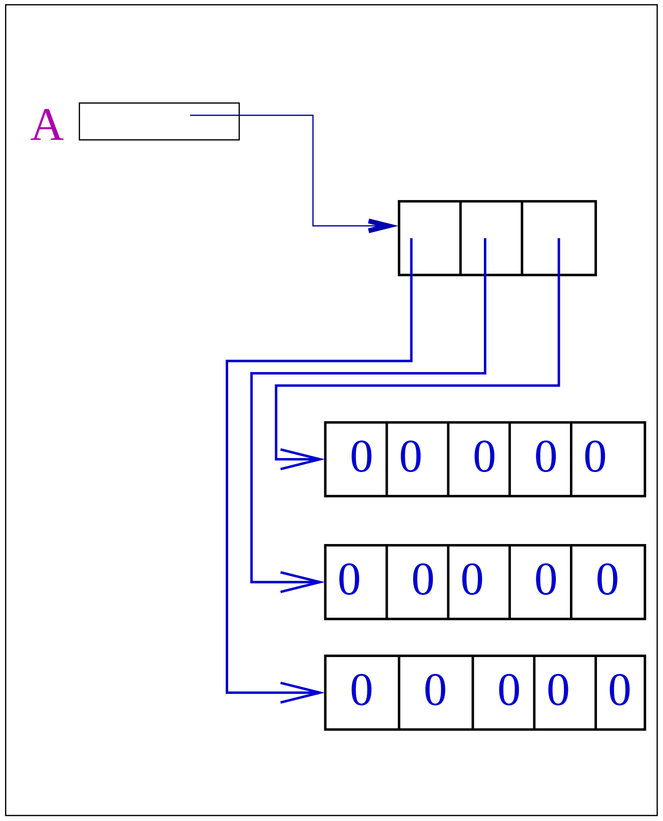
```
int[ ][ ] A ;

A  = new int[3][ ];

A [0] = new int[4];

A [1] = new int[2];

A [2] = new int[5];

 A [2][0] = 67;
```



Array of integer arrays with predefined size

```
int[ ][ ]  A ;

A  = new int[3][5];
```

Memory



Matrix : array of “arrays of same length”

1. How to add two matrices ?

   (Do it as exercise)

2. How to multiply two matrices ?

   (Code available on homepage : matrix_mul.java)

Characters : ’a’,’%’,’@’,...

1. *Unicode* defines a character set that can represent all of characters found in Human languages.
2. Java uses *Unicode*.

   char is an unsigned 16-bit type in the range 0 to 65,535. such that each character has a unique code in this range
3. unicode of ’a’ is 97.
4. unicode of ’z’ is 122.

Characters

- unicode of ’A’ to ’Z’ is from 65–90.
- unicode of ’a’ to ’z’ is from 97–122.
- unicode of ’0’ to ’9’ is 48–57.

Please go through the following simple programme to print characters with unicode from 0 to 127.

character_example2.java

Characters can be manipulated like integers

```
char firstletter = 97;
int i = firstletter;
char secondletter = (char)(firstletter+1);
System.out.println(firstletter);
System.out.println(i);
System.out.println(secondletter);
```

Output :

Characters can be manipulated like integers

```
char firstletter = 97;
int i = firstletter;
char secondletter = (char)(firstletter+1);
System.out.println(firstletter);
System.out.println(i);
System.out.println(secondletter);
```

Output :

```
a
97
b
```

Characters can be manipulated like integers

Please go through the following programme provided on the course website. They are very useful.

character_example1.java

String class

1. Its objects are sequence of characters. Example : “abc”, “world”,...
2. **Constructors :**

   - String() :

     It creates an empty string

   - String(String value) : It ceates a copy of the string referenced by value

In addition : there is a direct way to create a string.

String s = “my_name”;

Operations available on String objects

Let **str** be a reference to a string object

- `str.length()` : length of the string
- `str.charAt(i)` : the character at position `i`
- `str.indexOf(charc)` : returns the first position where charc appears and -1 if it does not appear.

Comparing two String objects

`str.compareTo(anotherString)`

returns **int** which is less than or equal to or greater than 0 if the String str is less than or equal to or greater than string anotherString.

Note that the strings are compared lexicographically (like in dictionary). So

- “Z” is less than “ABC”
- “Y” is less than “YA”;
- “abc” is less than “aabcdef”

What if there are special characters ?

Comparing two string obects

General algorithm :

Let **s** and **t** are two strings to be compared.

- start scanning **s** and **t** from left to right character by character until you find a mismatch. The order (less than or greater than) between the two strings is determined by the comparison between the unicode of the two characters.