ऐरे क्या है?

ऐरे, समान डाटा टाइप के वेरिएबल्स का समूह होता हैं। ऐरे, समान डाटा टाइप्स के वेरिएबल्स को स्टोर करने के लिए इस्तेमाल किया जाने वाला एक कंटेनर ऑब्जेक्ट हैं।

ऐरे परिभाषित (डिक्लेअर) करते समय ऐरे तय साइज़ देना जरुरी हैं। कंप्यूटर के मेमोरी में ऐरे साइज़ का कंटिन्यूअस ब्लॉक ऑपरेटिंग सिस्टम द्वारा आबंटित किया जाता हैं।

ऐरे के हर सदस्य वेरिएबल को एलिमेंट कहा जाता हैं और हर एलिमेंट का एक इंडेक्स (अनुक्रम) होता हैं, जिसका उपयोग कर ऐरे एलिमेंट को एक्सेस किया जा सकता हैं। ऐरे एलिमेंट की शुरुवात 0 से होती हैं।

उदाहरण के तौर पर, अगर ऐरे का साइज़ 10 डिक्लेअर किया गया हो तो उसमें 0 से 9 इंडेक्स तक, ऐरे एलीमेंट्स स्टोर किए जा सकते हैं।

ऐरे कैसे डिक्लेअर करें (How to declare array)

ऐरे डिक्लेअर करने के लिए सिंटेक्स-

Data-Type Array_name[];

उदाहरण:

int number[];

float average[];

उपर दी गए उदाहरणों में इन्टिजर और फ्लोट डाटा टाइप के ऐरे डिक्लेअर किए गए हैं।

ऐरे कैसे तयार करें (How to create array)

जावा में ऐरे तयार करने हेतु new ऑपरेटर का इस्तेमाल किया जाता हैं, new ऑपरेटर की मदत से ऐरे को मेमोरी स्पेस अलोकेट(आबंटित) की जाती हैं।

सिंटेक्स

Array-Name= new type[size];

उदाहरण

number= new int[6];

यहाँ, number यह ऐरे का नाम हैं जिसका डाटा टाइप इन्टिजर हैं, और इसकी साइज़ 6 हैं।

ऐरे को इनिशियलाइज़ कैसे करें (How to initialize array

इनिशियलाइजेशन का अर्थ होता हैं, ऐरे वेरिएबल को वैल्यू पास करना(वेरिएबल में वैल्यू स्टोर करना)। जब हम ऐरे डिक्लेअर करते हैं, उसमें ऑपरेटिंग सिस्टम द्वारा कोई भी गार्बेज वैल्यू स्टोर की जाती हैं।

जब हम ऐरे वेरिएबल में यूजर द्वारा इंटर की गयी वैल्यू स्टोर की जाती हैं, उसे प्रक्रिया को इनिशियलाइजेशन कहा जाता हैं।

प्रोग्राम एक्सीक्यूशन में, ऐरे वेरिएबल में यूजर द्वारा इंटर की गयी वैल्यू का ही इस्तेमाल किया जाता हैं।

सिंटेक्स

Array-Name[subscript/index]=value;

उदाहरण

number new= int [4];

प्रकार 1

number[0]=10;

number[1]=20;

number[2]=30;

number[3]=40

प्रकार 2

number[]={10,20,30,40};

ऐरे के प्रकार (Types of arrays

जावा में के ऐरे के दो प्रकार हैं

1. वन डायमेंशनल ऐरे

2. टू डायमेंशनल ऐरे

1. वन डायमेंशनल ऐरे (One dimensional array)

वन डायमेंशनल ऐरे, समान डाटा टाइप के वेरिएबल्स को एक पंक्ति(row) में स्टोर करता हैं। ऐरे का जनरल सिंटेक्स और वन डायमेंशनल ऐरे का सिंटेक्स दोनों समान हैं।

सिंटेक्स

Array-Name[subscript/index]=value;

उदाहरण

```
Class OneDArray
{
public static void main(String args[])
{
int a[] new= int[5];
a[0]=11
a[1]=22;
a[2]=33;
a[3]=44;
a[5]=55;
//to print array elements
for(i=0; i<a.length; i++)
//length is a property of an array
System.out.println(a[i]);
}
}
```

2. टू डायमेंशनल ऐरे (Two dimensional array)

वन डायमेंशनल ऐरे में ऐरे एलेमेंट्स, एक रो(पंक्ति) में स्टोर किए जाते हैं, इसके विपरीत टू डायमेंशनल ऐरे में ऐरे एलेमेंट्स मैट्रिक्स(रो-कॉलम्स) के रूप में स्टोर किए जाते हैं।

मैट्रिक्स फॉर्मेट में स्टोर किए जाने का अर्थ हैं, हर ऐरे एलिमेंट का एक रो इंडेक्स और एक कॉलम इंडेक्स होगा, उसकी मदत से किसी भी विशिष्ट ऐरे एलिमेंट को एक्सेस किया जा सकता हैं।

सिंटेक्स

Type array_name=new type[rows][columns];

उदाहरण

```
class TwoDArray

{

final static int r=3;

final static int c=3;

public static void main(String args[])

{

Int mult[] []= new int [r][c];

int row, columns;

int i, j;

for(i=1; i<=r; i++)

{

for(j=1; j<=c; j++)

{

mult[i] [j]= i*j;

System.out.println(“ “+mult[i][j]);

}

}

System.out. println(“ “);

}
```

}

STRING

String Java के लिए एक 'Non-primitive' Data Type होता है, स्ट्रिंग एक से अधिक characters का group होता है, स्ट्रिंग को हम double quotes(" ") के अन्दर लिखते है स्ट्रिंग को class; java.lang इस package में define किया जाता है. दोस्तों C और C++ में स्ट्रिंग के लिए character डाटा टाइप और array का इस्तेमाल किया जाता था. लेकिन आपको पता होना चाहिए, की Java में स्ट्रिंग के object को create किया जा सकता है।

Java String क्लास स्ट्रिंग पर ऑपरेशन करने के लिए कई methods प्रदान करता है जैसे compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() आदि, Java में भी C और C++ के जैसे string को भी create किया जाता है।

String literal

Java string को शाब्दिक डबल quotes का उपयोग करके बनाया गया है, उदाहरण के लिए string s = "Hello";

Using new keyword

Java string को एक कीवर्ड "new" का उपयोग करके बनाया गया है, उदाहरण के लिए string s = new string ("Hello");

Example

```
public class Sample {


    public static void main(String args[]) {
        String str1 = "Hello ", str2 = "World!";
        System.out.println(str1.concat(str2));
    }
}
```

Result

"Hello," + " world" + "!"

रैपर क्लास (Wrapper class i)

जैसा की आप जानते हैं, जावा एक ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग लैंग्वेज हैं। इसका मतलब जावा में सभी प्रोग्राम क्लास और उस क्लास के ऑब्जेक्ट्स पर आधारित होते हैं। जावा प्रोग्रामिंग में प्रिमिटिव डाटा टाइप को इस्तेमाल करने हेतु उनका ऑब्जेक्ट तयार करना जरुरी होता हैं, लेकिन ऑब्जेक्ट क्रिएट करने के लिए **क्लास** का होना भी जरुरी हैं।

इसी समस्या को दूर करने के लिए जावा में रैपर क्लास की सुविधा उपलब्ध करायी गयी हैं। रॅपर क्लास इन प्रिमिटिव डेटा टाइप्स जैसे- इन्टिजर, करैक्टर, फ्लोट, बाइट, डबल इत्यादि को क्लास में **एनकैप्सूलेट** करता हैं, जिस क्लास का ऑब्जेक्ट तयार किया जा सकें।

दुसरे शब्दों में रॅपर क्लास प्रिमिटिव डेटा टाइप्स को एक क्लास में **रॅप** करता हैं, जिससे उन प्रिमिटिव डेटा टाइप्स का इस्तेमाल आसानी से किया जा सकें।

रैपर क्लास की जरुरत (Need of wrapper classes in Hindi)

1. **रैपर क्लास** प्रिमिटिव डाटा टाइप्स को ऑब्जेक्ट्स में कन्वर्ट करता हैं।
2. Java.util इस इन बिल्ड पैकेज में लिखे गए क्लासेज सिर्फ और सिर्फ क्लास ऑब्जेक्ट को ही हैंडल कर सकते हैं, इसी लिए रैपर क्लास का इस्तेमाल किया जाता हैं।
3. अगर प्रिमिटिव डेटा टाइप्स को रैपर क्लास में **रॅप** नहीं किया जाएगा तो, उन डेटा टाइप्स और उन्हें पास किए गए वैल्यूज का इस्तेमाल करना मुश्किल हो जाएगा।
4. **जावा प्रोग्रामिंग** में इस्तेमाल किए जानेवाले **डेटा स्ट्रक्चर्स** जैसे की ऐरे लिस्ट, वेक्टर में सिर्फ क्लास ऑब्जेक्ट्स को स्टोर किया जा सकता हैं, न की प्रिमिटिव डाटा टाइप्स को, इसीलिए रॅपर क्लास का इस्तेमाल किया जाता हैं।
5. **मल्टी थ्रेडिंग सिंक्रोनाइजेशन** सिर्फ क्लास ऑब्जेक्ट को सपोर्ट करता हैं, इसलिए क्लास का होना जरुरी बन जाता हैं।

आसान शब्दों में, जावा प्रोग्राम में किसी भी प्रकार के ऑपरेशन का परफॉर्म करने के लिए क्लास की जरुरत होती हैं। क्योंकि प्रिमिटिव डेटा टाइप्स को प्रोग्राम में इस्तेमाल करने के लिए भी एक क्लास की जरुरत हैं, इसी लिए रैपर क्लास का इस्तेमाल किया जाता हैं। रैपर क्लास की मदत से प्रिमिटिव डेटा टाइप्स का भी ऑब्जेक्ट तयार किया जा सकता हैं।

प्रिमिटिव डेटा टाइप्स और रैपर क्लास (Primitive data types and corresponding wrapper classes)

प्रिमिटिव डेटा टाइप्स	रैपर क्लास

byte	**Byte**
short	**Short**
int	**Integer**
long	**Long**
float	**Float**
double	**Double**
char	**Character**
boolean	**Boolean**

ऑटोबॉक्सिंग और अनबॉक्सिंग (Autoboxing and Unboxing)

ऑटोबॉक्सिंग (Autoboxing)

प्रिमिटिव डेटा टाइप्स का उनके निर्धारित रैपर क्लास में किए गए आटोमेटिक कन्वर्शन को ऑटोबॉक्सिंग कहा जाता हैं। जैसे की int इस प्रिमिटिव डेटा टाइप का कन्वर्शन (रूपांतरण) Integer इस रॅपर क्लास में किया जाना।

उदाहरण-

```
class Boxing

{

public static void main(String args[])

{

int a=50;

Integer a2=new Integer(a);

System.out.println(a2);

}

}
```

आउटपुट-

50

स्पष्टीकरण-

ऑटोबॉक्सिंग को समझने के लिए हमने ऊपर एक आसान प्रोग्राम लिखा हैं। हमने एक क्लास डिक्लेअर किया हैं, Boxing। क्लास Boxing के मेन फंक्शन में हमने, एक इन्टिजर a डिक्लेअर किया हैं, जिसे '50' यह वैल्यू असाइन की हैं।

अब इन्टिजर इस प्रिमिटिव डेटा टाइप के जुड़े रॅपर क्लास जिसका नाम **इन्टिजर** हैं, का ऑब्जेक्ट क्रिएट किया गया हैं, और उसके कंस्ट्रक्टर को इन्टिजर वेरिएबल a पैरामीटर के तौर पर पास किया हैं। इन्टिजर क्लास के ऑब्जेक्ट a2 को इनिशियलाइज़ करने हेतु इन्टिजर वेरिएबल a को पास किया हैं।

अब, जब प्रिंटिंग फंक्शन में a2 को कॉल किया जाएगा तब, वह ऑब्जेक्ट इन्टिजर वेरिएबल की वैल्यू रीटर्न करेगा, जोकि उसे पैरामीटर के रूप में पास किए गयी हैं।

अनबॉक्सिंग (Unboxing in Hindi)

रॅपर क्लास के ऑब्जेक्ट को प्रिमिटिव डेटा टाइप में कन्वर्ट किए जाने को अनबॉक्सिंग कहा जाता हैं। अनबॉक्सिंग , ऑटोबॉक्सिंग बिलकुल विपरीत हैं। जो कार्य ऑटोबॉक्सिंग में किया जाता हैं, उसके विपरीत काम अनबॉक्सिंग में किया जाता हैं।

उदाहरण-

```
class Unboxing

{

public static void main(String args[])

{

Integer i=new Integer(50);

int a=i;


System.out.println(a);

}

}
```

आउटपुट-

```
50
```

स्पष्टीकरण-

जैसा की आप अब जान चुकें हैं, की अन बॉक्सिंग, ऑटोबॉक्सिंग के बिलकुल उल्टा हैं। ऊपर दिए गए उदाहरण में एक क्लास लिया हैं, Unboxing। इस क्लास के मेन फंक्शन में हमने इन्टिजर क्लास का एक ऑब्जेक्ट तयार किया हैं 'i' और कंस्ट्रक्टर को वैल्यू पास की हैं 50, जोकि ऑब्जेक्ट में स्टोर की गयी हैं।

अब हमने इस इन्टिजर क्लास के ऑब्जेक्ट को इन्टिजर जोकि एक प्रिमिटिव डेटा टाइप हैं, उसके वेरिएबल में कन्वर्ट करना हैं। इसलिए हमने एक इन्टिजर वेरिएबल लिया हैं a और उसे असाइनमेंट ऑपरेटर(=) की मदत से, ऑब्जेक्ट i को असाइन किया हैं। अब इन्टिजर वेरिएबल a में ऑब्जेक्ट i की वैल्यू स्टोर की जा चुकी हैं।

प्रोग्राम के अंत में अनबॉक्सिंग हुयी हैं, की नहीं यह जांचने के लिए जब प्रिंट फंक्शन को कॉल किया हैं और उसे a यह वेरिएबल पास किया हैं। अब जावा कम्पाइलर a की वैल्यू 50 यह प्रिंट करेगा।

सारांश(Brief summary of wrapper classes in java)

रॅपर क्लास प्रिमिटिव डेटा टाइप्स को एक क्लास में **रॅप** करता हैं, जिस क्लास का ऑब्जेक्ट तयार किया जा सकें। और उन प्रिमिटिव डेटा टाइप्स का इस्तेमाल आसानी से किया जा सकें।

अगर आप ठीक से देखें तो-

ऑटो बॉक्सिंग-

प्रिमिटिव डेटा टाइप को रॅपर क्लास के ऑब्जेक्ट में कन्वर्ट करता हैं।

अनबॉक्सिंग

रॅपर क्लास के ऑब्जेक्ट को प्रिमिटिव डेटा टाइप में कन्वर्ट करता हैं।