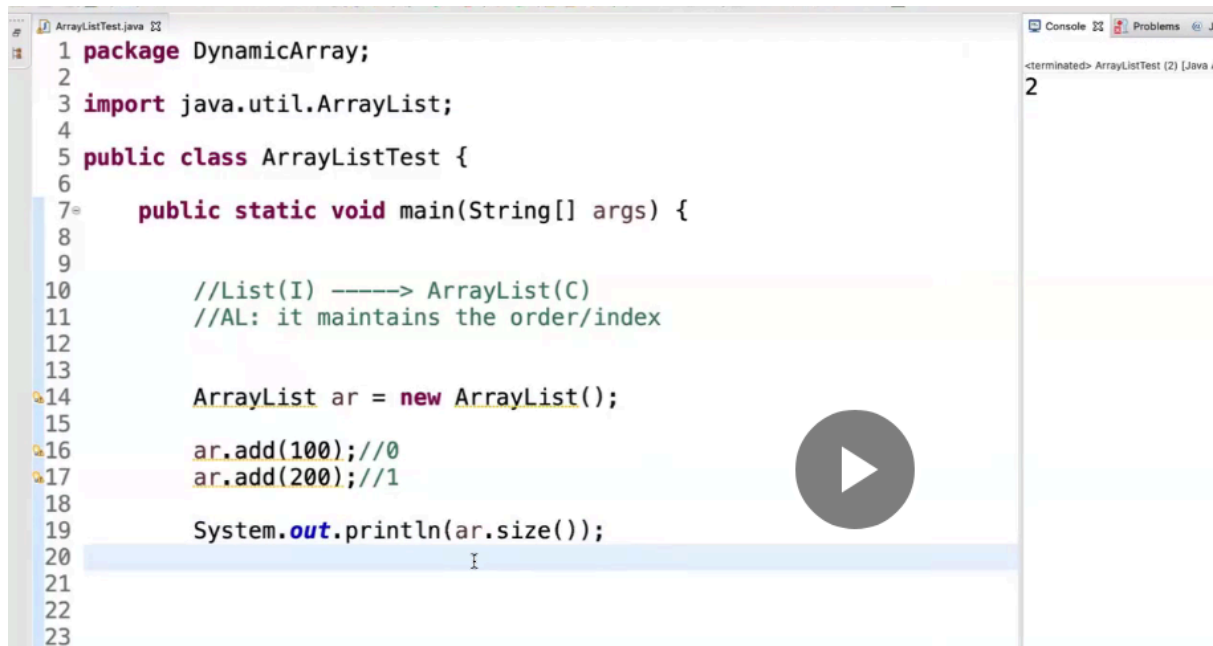


1



```
1 package DynamicArray;
2
3 import java.util.ArrayList;
4
5 public class ArrayListTest {
6
7     public static void main(String[] args) {
8
9
10         //List(I) ----> ArrayList(C)
11         //AL: it maintains the order/index
12
13
14         ArrayList ar = new ArrayList();
15
16         ar.add(100); //0
17         ar.add(200); //1
18
19         System.out.println(ar.size());
20
21
22
23
```

```
20
21         ar.add(300); //2
22         ar.add(400); //3
23
24         System.out.println(ar.size()); //4
25
```

4

```
13
14         ArrayList ar = new ArrayList();
15         System.out.println(ar.size()); //0
```

0

~~8.00~~

10 virtual segments created for array list by default in heap.

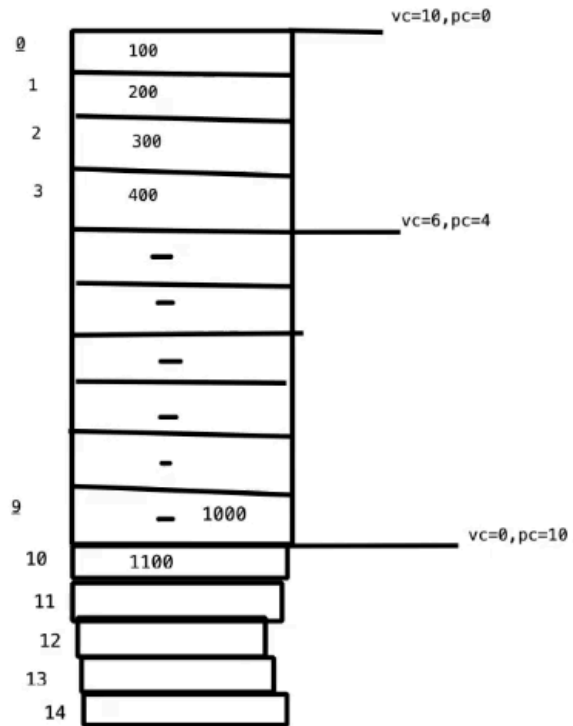
Physical capacity is depending on how many elements present. All methods on array list operate on physical capacity.

No method to check virtual capacity.

```

ArrayList ar = new ArrayList();
default vc = 10
+
0 to 9: 10 values are filled
[0] to [9]: full
[10]: 11th value: ar.add(1100)
LF = pc/2 = 10/2 = 5 (vc)

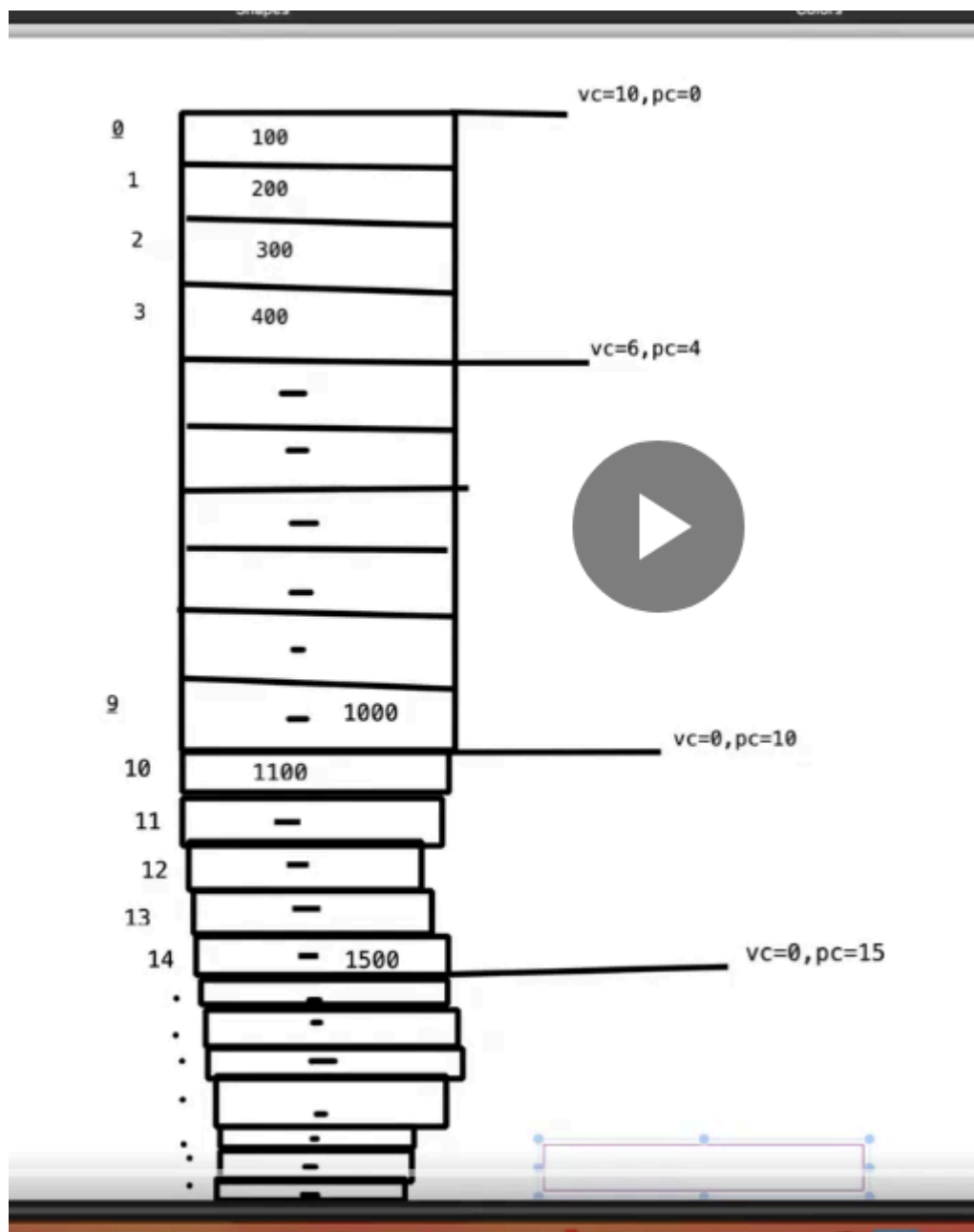
```



```

[10] to [14]: filled
[15]: 16th value:
LF = pc/2=15/2=7 (vc)

```



paste arrlist1

```

arrlist1.java X
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist1 {
6
7     public static void main(String[] args) {
8
9         ArrayList a1=new ArrayList();
10        int i1=a1.size();
11        System.out.println(i1);//0
12        a1.add(100);
13        a1.add(200);
14        int size = a1.size();
15        System.out.println(size);
16
17        a1.add(300);
18        a1.add(400);
19        int size1=a1.size();
20        System.out.println(size1);
21
22        //add returns true or false.
23        boolean b1=a1.add(500);
24        System.out.println(b1);//true
25    }
26
27 }
28 //2
29 //4
30

```

We can specify default virtual capacity-

```

27
28 ArrayList ar1 = new ArrayList(5);//vc=5,pc=0
29
30

```

paste arrlist2

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist2 {
6
7     public static void main(String[] args) {
8
9         //specify default virtual capacity and print array list.
10        //we get [].
11        ArrayList a1=new ArrayList(50);
12        System.out.println(a1);
13    }
14
15 }
16 //[]
17
18

```

Remove-

Entire index removed and lower indexes shifted up.

```

1 package DynamicArray;
2
3 import java.util.ArrayList;
4
5 public class ArrayListPractice {
6
7     public static void main(String[] args) {
8
9         ArrayList ar = new ArrayList();//vc=10,pc=0
10
11        System.out.println(ar.size());//0
12
13        ar.add(100);//0
14        ar.add(200);//1
15        ar.add(300);//2
16        ar.add(400);//3
17
18        System.out.println(ar.size());//4
19
20        ar.remove(2);
21
22        System.out.println(ar.size());//3
23

```



paste arrlist3

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist3 {
6
7     public static void main(String[] args) {
8
9         ArrayList a1=new ArrayList();
10        int i1=a1.size();
11        System.out.println(i1);//0
12        a1.add(100);
13        a1.add(200);
14        int size = a1.size();
15        System.out.println(size); //2
16
17        a1.add(300);
18        a1.add(400);
19        int size1=a1.size();
20        System.out.println(size1); //4
21
22        //remove - pass index returns object.
23        Object o1=a1.remove(0); //pass in index.
24        System.out.println(o1); //100 - removed element seen.
25        int size2=a1.size();
26        System.out.println(size2);//3
27    }
28
29 }
30

```

Get-

Pass index.

```

19
20        System.out.println(ar.get(1));
21

```

200

paste [arrlist4](#)

```
arrlist4.java ×
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist4 {
6
7     public static void main(String[] args) {
8
9         ArrayList a1=new ArrayList();
10        int i1=a1.size();
11        System.out.println(i1); //0
12        a1.add(100);
13        a1.add(200);
14        int size = a1.size();
15        System.out.println(size); //2
16
17        a1.add(300);
18        a1.add(400);
19        int size1=a1.size();
20        System.out.println(size1); //4
21
22        //pass index and get the value.
23
24        Object object = a1.get(2);
25        System.out.println(object);
26    }
27
28 }
```

```
27
28 }
29
30 //0
31 //2
32 //4
33 //300
34
35
```

Remove based on index number-

```
21
22     ar.remove(1);
23
24     System.out.println(ar.get(1));
25
```

```
22     ar.remove(1);
23
24     System.out.println(ar.get(1));
25
```

300

Print all values-

```
26     //print all the value of ArrayList:
27
28     System.out.println(ar);
29
```

[100, 200, 300, 400]

Access out of index-

```
19
20     System.out.println(ar.get(3));
21     System.out.println(ar.get(4));
22
```

```
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:65)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:75)
```

paste [arrlist6](#)


```

arrlist6.java x
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist6 {
6
7     public static void main(String[] args) {
8
9         ArrayList a1=new ArrayList();
10        int i1=a1.size();
11        System.out.println(i1); //0
12        a1.add(100);
13        a1.add(200);
14        int size = a1.size();
15        System.out.println(size); //2
16
17        a1.add(300);
18        a1.add(400);
19        int size1=a1.size();
20        System.out.println(size1); //4
21
22        //try accessing from index not present.
23        //out of bounds exception.
24        Object object = a1.get(101);
25        System.out.println(object);
26    }
27
28 }
29
30 //0
31 //2
32 //4
33 //Exception in thread "main" java.lang.IndexOutOfBoundsException:
34 // Index 101 out of bounds for length 4
35 // at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:100)
36 // at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
37 // at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
38 // at java.base/java.util.Objects.checkIndex(Objects.java:365)
39 // at java.base/java.util.ArrayList.get(ArrayList.java:428)
40 // at com.day28.arrlist6.main(arrlist6.java:22)
41
42

```

Print values one by one-

```

34 //use for loop:
35 //index loop:
36 for(int i=0; i<ar.size(); i++) {
37     System.out.println(ar.get(i)); //100 200 300 400
38 }
39

```

```

100
200
300
400

```

Storing any value-

```

44
45 //
46 ArrayList ls = new ArrayList();
47 ls.add(100);
48 ls.add(12.33);
49 ls.add("testing");
50 ls.add(true);
51 ls.add('a');
52
53 System.out.println(ls);
54

```

```

100
[100, 12.33, testing, true, a]

```

Generics-

Cant add any other value.

```

55 //ArrayList with Generics:
56
57 ArrayList<Integer> numList = new ArrayList<Integer>(); //vc=10, pc=0
58 numList.add(100);
59 numList.add(200);
60 numList.add("testing");
61
62
63
64
65 }
66

```

The method add(Integer) in the type ArrayList<Integer> is not applicable for the arguments (String)

1 quick fix available:

Change to 'addAll()'

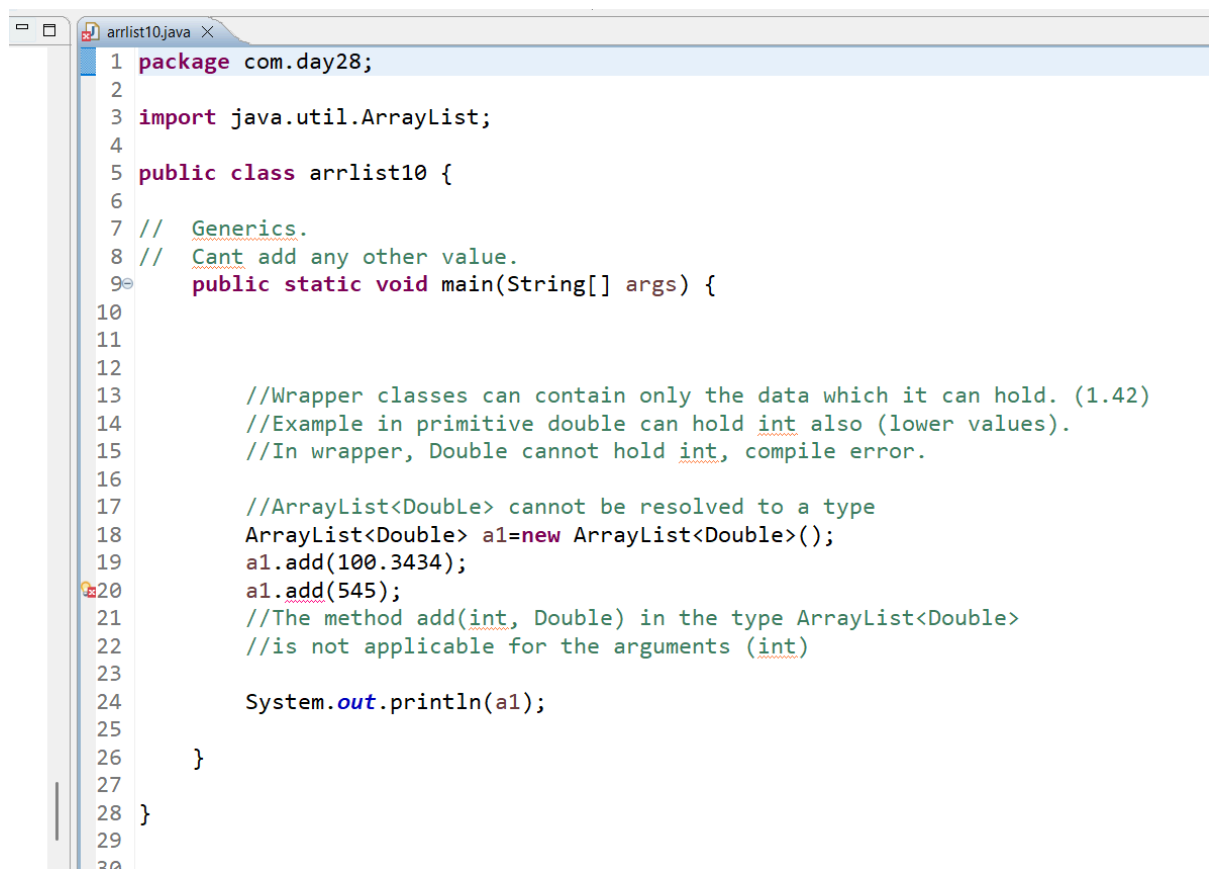
Double cannot hold integer-

```

61
62     ArrayList<Double> marksList = new ArrayList<Double>();//vc=10, pc=0
63     marksList.add(12.33);
64     marksList.add(200);
65

```

paste [arrlist10](#)



```

1  package com.day28;
2
3  import java.util.ArrayList;
4
5  public class arrlist10 {
6
7      // Generics.
8      // Cant add any other value.
9      public static void main(String[] args) {
10
11
12
13         //Wrapper classes can contain only the data which it can hold. (1.42)
14         //Example in primitive double can hold int also (lower values).
15         //In wrapper, Double cannot hold int, compile error.
16
17         //ArrayList<Double> cannot be resolved to a type
18         ArrayList<Double> a1=new ArrayList<Double>();
19         a1.add(100.3434);
20         a1.add(545);
21         //The method add(int, Double) in the type ArrayList<Double>
22         //is not applicable for the arguments (int)
23
24         System.out.println(a1);
25
26     }
27
28 }
29
30

```

String-

```

66
67     ArrayList<String> browserList = new ArrayList<String>();//vc=10, pc=0
68     browserList.add(100);
69

```

paste [arrlist11](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist11 {
6
7     // Generics.
8     // Cant add any other value.
9     public static void main(String[] args) {
10
11         ArrayList<String> a1=new ArrayList<String>();
12         a1.add("tiger");
13         a1.add(32424);
14         //The method add(int, String) in the type ArrayList<String>
15         //is not applicable for the arguments (int)
16         System.out.println(a1);
17     }
18 }
19
20 }
21

```

```

55 //ArrayList with Generics:
56
57 ArrayList<Integer> numList = new ArrayList<Integer>();//vc=10, pc=0
58 numList.add(100);
59 numList.add(200);
60
61
62 ArrayList<Double> marksList = new ArrayList<Double>();//vc=10, pc=0
63 marksList.add(12.33);
64 marksList.add(200.00);
65
66
67 ArrayList<String> browserList = new ArrayList<String>();//vc=10, pc=0
68 browserList.add("chrome");
69 browserList.add("firefox");
70 browserList.add("edge");
71

```

paste arrlist12

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist12 {
6
7     // Generics.
8     // Cant add any other value.
9     public static void main(String[] args) {
10
11         //Different types of generics.
12         ArrayList<Integer> a1=new ArrayList<Integer>();
13         a1.add(34234);
14         a1.add(20);
15         System.out.println(a1);
16
17         ArrayList<Double> a2=new ArrayList<Double>();
18         a2.add(34234.3245435);
19         a2.add(20.54578);
20         System.out.println(a2);
21
22         ArrayList<String> a3=new ArrayList<String>();
23         a3.add("lion");
24         a3.add("gorialla@#$324");
25         System.out.println(a3);
26
27     }
28

```

```

26
27     }
28
29 }
30
31 //[34234, 20]
32 //[34234.3245435, 20.54578]
33 //[lion, gorialla@#$324]
34

```

Object type-

```

73 ArrayList<Object> empDataList = new ArrayList<Object>();//vc=10, pc=0
74 empDataList.add("Tom");
75 empDataList.add(30);
76 empDataList.add(12.33);
77 empDataList.add('m');
78 empDataList.add(true);
79

```

paste arrlist13

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist13 {
6
7     //object type generics.
8     public static void main(String[] args) {
9
10        //can store any data type.
11        //object type generics.
12        ArrayList<Object> a1=new ArrayList<Object>();
13        a1.add(56.87);
14        a1.add(20);
15        a1.add('t');
16        a1.add(true);
17        a1.add("tiger");
18        System.out.println(a1);
19
20    }
21
22 }
23
24 //[56.87, 20, t, true, tiger]
25
26

```

Right side is not mandatory but if we write no harm

```

72
73 ArrayList<Object> empDataList = new ArrayList<>();//vc=10, pc=0
74 empDataList.add("Tom");
75 empDataList.add(30);
76 empDataList.add(12.33);
77 empDataList.add('m');
78 empDataList.add(true);
79

```

```

78      empDataList.add(empData);
79
80      System.out.println(empDataList);
81
82      [100, 12.33, testing, true, d]
83      [Tom, 30, 12.33, m, true]

```

For each loop-

```

65
66      System.out.println("-----");
67
68      ArrayList<String> browserList =
69      browserList.add("chrome");
70      browserList.add("firefox");
71      browserList.add("edge");
72
73      for(String e : browserList) {
74          System.out.println(e);
75      }
76

```

Chrome

Ff

Edge

With loops under loops in for each-

```

72
73      for(String e : browserList) {
74          System.out.println(e);
75          if(e.equals("firefox")) {
76              System.out.println("enter url");
77          }
78      }
79

```

Chrome

Ff

Enter url

Edge

paste [arrlist15](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist15 {
6
7     //object type generics.
8     public static void main(String[] args) {
9
10        //loops under loops.
11        ArrayList<Object> a1=new ArrayList<Object>();
12        a1.add(56.87);
13        a1.add(20);
14        a1.add('t');
15        a1.add(true);
16        a1.add("tiger");
17
18        for(Object e:a1) {
19            System.out.println(e);
20            if(e.equals(20)) {
21                System.out.println("enter url.");
22                break;
23            }
24            return 10; //Void methods cannot return a value
25        }
26    }
27 }

```

```

28
29 }
30
31 //56.87
32 //20
33 //enter url.
34
35

```


With break-

```
72  
73     for(String e : browserList) {  
74         System.out.println(e);  
75         if(e.equals("firefox")) {  
76             System.out.println("enter url");  
77             break;  
78         }  
79     }  
80
```

Chrome

Ff

Enter url

Print object-

```
92  
93     for(Object e : empDataList) {  
94         System.out.println(e);  
95     }  
96
```

```
Tom  
30  
12.33  
m  
true
```

Duplicate value-

Allowed.

```

96
97 //
98 ArrayList<String> studentList = new ArrayList<String>();//vc=10, pc=0
99
100 studentList.add("monika");//0
101 studentList.add("sunil");//1
102 studentList.add("vibha");//2
103 studentList.add("surya");//3
104 studentList.add("sunil");//4
105
106 System.out.println(studentList);
107

```

```

true
[monika, sunil, vibha, surya, sunil]

```

```

107 studentList.remove(4);
108 System.out.println(studentList);
109

```

```

[monika, sunil, vibha, surya]

```

paste arrlist19

```

arrlist19.java X
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist19 {
6
7     //String type generics.
8     public static void main(String[] args) {
9
10        //remove student with index number.
11        ArrayList<String> studentnames=new ArrayList<String>();
12        studentnames.add("james");
13        studentnames.add("vibha");
14        studentnames.add("bond");
15        studentnames.add("james");
16
17        String s1=studentnames.remove(2);
18        System.out.println(s1);
19        System.out.println(studentnames);
20    }
21
22 }
23
24 //bond
25 //[james, vibha, james]
26

```

Null-
Allowed.

```
104 studentList.add("sunil");
105 studentList.add(null);
106

[monika, sunil, vibha, surya, sunil, null]
```

Multiple null-

```
99
100 studentList.add(null);
101 studentList.add("monika");//0
102 studentList.add("sunil");//1
103 studentList.add("vibha");//2
104 studentList.add("surya");//3
105 studentList.add("sunil");//4
106 studentList.add(null);
107 studentList.add(null);
108
109
```

Allowed.

```
[null, monika, sunil, vibha, surya, sunil, null, null]
```

Back to back duplicates-

```
99
100 studentList.add(null);
101 studentList.add("monika");//0
102 studentList.add("monika");//0
103 studentList.add("sunil");//1
104 studentList.add("vibha");//2
105 studentList.add("surya");//3
106 studentList.add("sunil");//4
107 studentList.add(null);
108 studentList.add(null);
109

[null, monika, monika, sunil, vibha, surya, sunil, null, null]
```

Integer non primitive-

Null allowed.

```

50
57     ArrayList<Integer> numList = new ArrayList<Integer>();//vc=10, pc=0
58     numList.add(100);
59     numList.add(200);
60     numList.add(null);
61

```

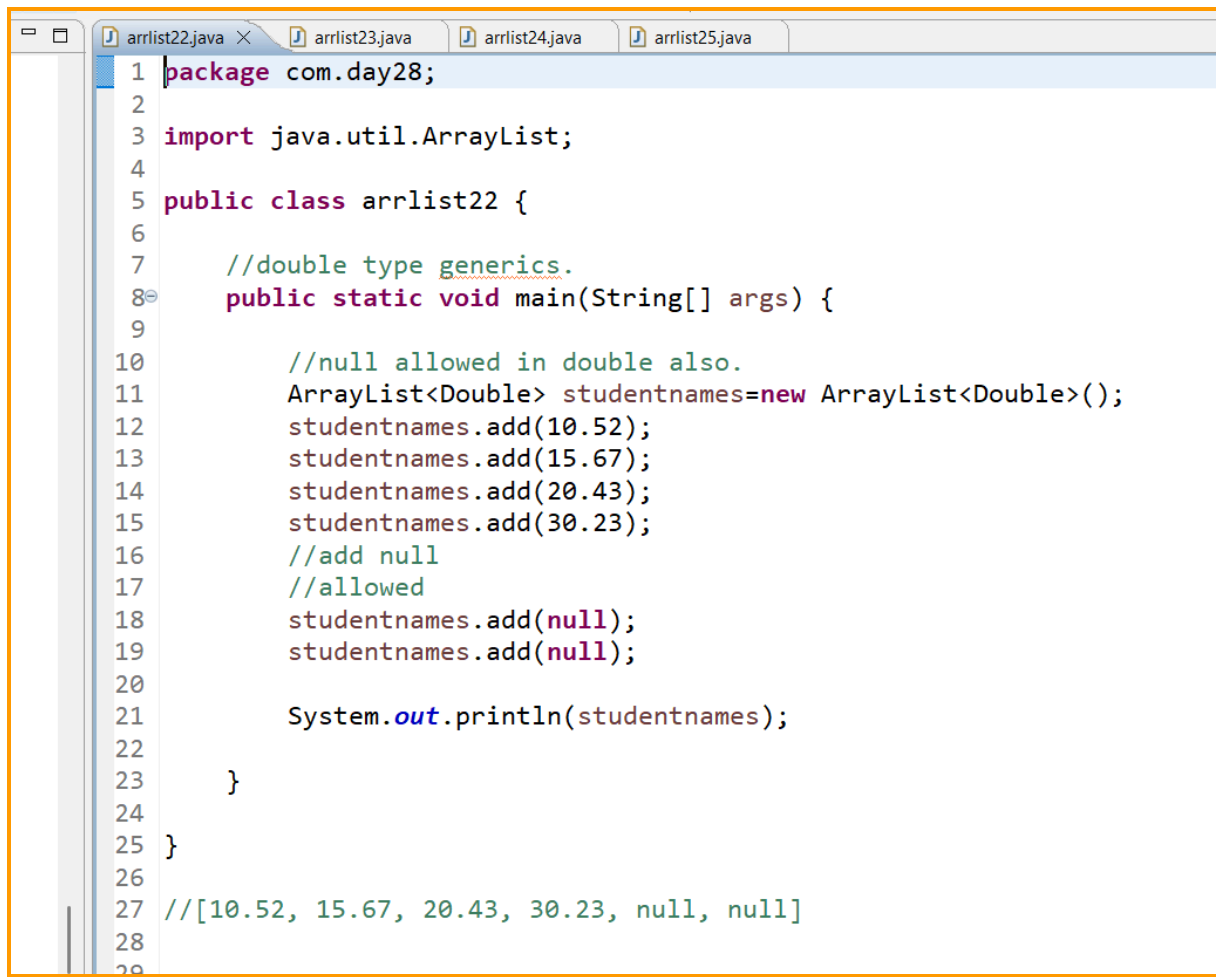
paste arrlist21

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist21 {
6
7     //integer type generics.
8     public static void main(String[] args) {
9
10        //null allowed in integer also.
11        ArrayList<Integer> studentnames=new ArrayList<Integer>();
12        studentnames.add(10);
13        studentnames.add(15);
14        studentnames.add(20);
15        studentnames.add(30);
16        //add null
17        //allowed
18        studentnames.add(null);
19        studentnames.add(null);
20
21        System.out.println(studentnames);
22
23    }
24
25 }
26
27 //[10, 15, 20, 30, null, null]
28
29

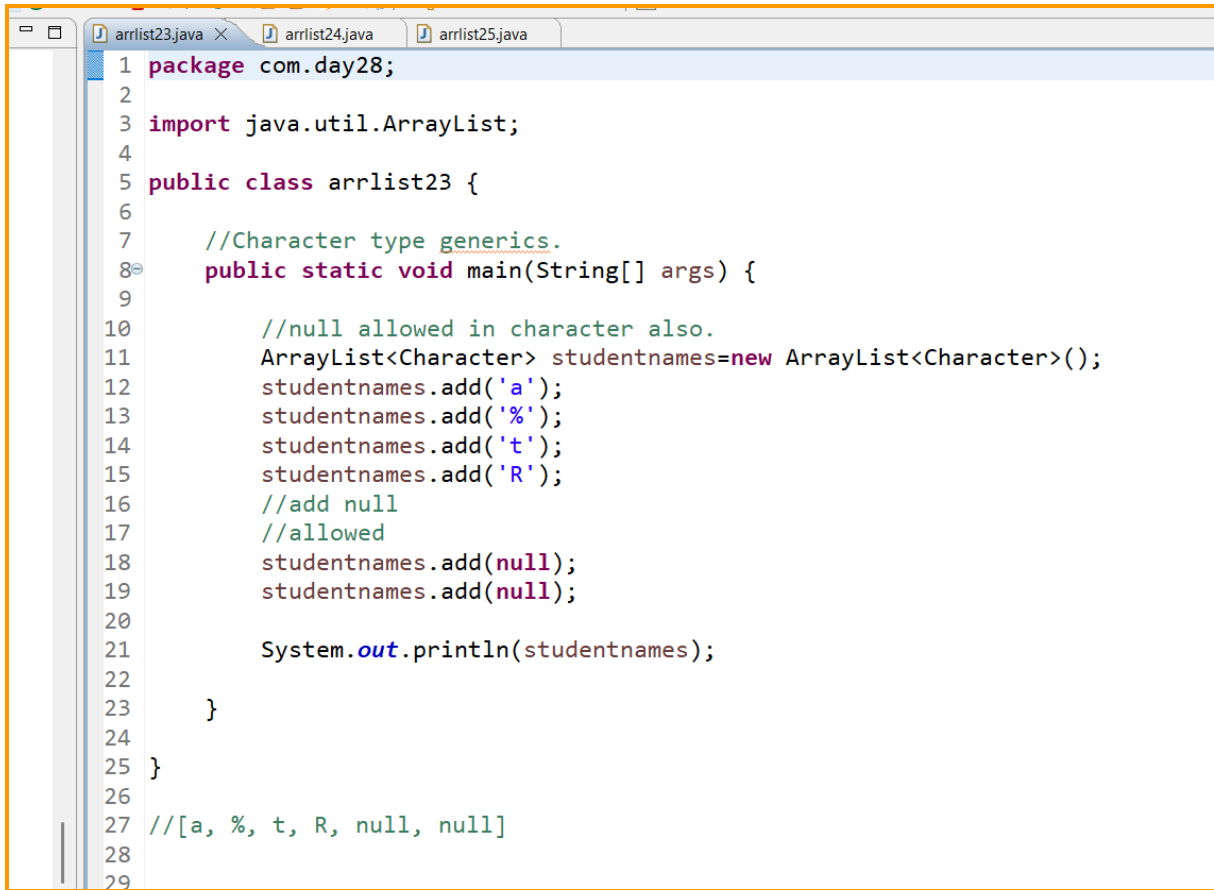
```

paste arrlist22



```
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist22 {
6
7     //double type generics.
8     public static void main(String[] args) {
9
10        //null allowed in double also.
11        ArrayList<Double> studentnames=new ArrayList<Double>();
12        studentnames.add(10.52);
13        studentnames.add(15.67);
14        studentnames.add(20.43);
15        studentnames.add(30.23);
16        //add null
17        //allowed
18        studentnames.add(null);
19        studentnames.add(null);
20
21        System.out.println(studentnames);
22
23    }
24
25 }
26
27 //[10.52, 15.67, 20.43, 30.23, null, null]
```

paste arrlist23



```
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist23 {
6
7     //Character type generics.
8     public static void main(String[] args) {
9
10        //null allowed in character also.
11        ArrayList<Character> studentnames=new ArrayList<Character>();
12        studentnames.add('a');
13        studentnames.add('%');
14        studentnames.add('t');
15        studentnames.add('R');
16        //add null
17        //allowed
18        studentnames.add(null);
19        studentnames.add(null);
20
21        System.out.println(studentnames);
22    }
23 }
24
25 }
26
27 //[a, %, t, R, null, null]
28
29
```

paste arrlist24

```
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist24 {
6
7     //Float type generics.
8     public static void main(String[] args) {
9
10        //null allowed in float also.
11        ArrayList<Float> studentnames=new ArrayList<Float>();
12        studentnames.add(10.25f);
13        studentnames.add(58.98f);
14        studentnames.add(45.56F);
15        studentnames.add(67.89F);
16        //add null
17        //allowed
18        studentnames.add(null);
19        studentnames.add(null);
20
21        System.out.println(studentnames);
22
23    }
24 }
25
26
27 //[10.25, 58.98, 45.56, 67.89, null, null]
28
29
```

paste arrlist25

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist25 {
6
7     //Boolean type generics.
8     public static void main(String[] args) {
9
10        //null allowed in boolean also.
11        ArrayList<Boolean> studentnames=new ArrayList<Boolean>();
12        studentnames.add(true);
13        // studentnames.add(True);//True cannot be resolved to a variable
14        // studentnames.add(False);//False cannot be resolved to a variable
15        // studentnames.add(TRUE);//TRUE cannot be resolved to a variable
16        // studentnames.add(FALSE);//FALSE cannot be resolved to a variable
17        studentnames.add(false);
18        //add null
19        //allowed
20        studentnames.add(null);
21        studentnames.add(null);
22
23        System.out.println(studentnames);
24    }
25 }
26
27 }
28
29 //[true, false, null, null]
30

```

```

117 //
118 ArrayList<String> footerList = new ArrayList<String>();//vc=10, pc=0
119 footerList.add("contact us");
120 footerList.add("help");
121 footerList.add("delivery info");
122 footerList.add("Returns");
123 footerList.add("cart");
124 footerList.add("accounts");
125
126
127 for(String e : footerList) {
128     System.out.println(e);
129     if(e.equals("delivery info")) {
130         System.out.println("click on it");
131         break;
132     }
133 }
134
135

```

```

contact us
help
delivery info
t click on it

```



Generics cannot be of primitive data types.(1.07)
and (1.15)

Order based collection-

```

117
118
119 //
120 ArrayList<String> footerList = new ArrayList<String>(30); //vc=30, pc=0
121 footerList.add("contact us");//0
122 footerList.add("help");//1
123 footerList.add("delivery info");//2
124 footerList.add("Returns");//3
125 footerList.add("cart");//4
126 footerList.add("accounts");//5
127 footerList.add(8, "Naveen");//IndexOutOfBoundsException

```



We cant randomly add at any index.

```

Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 8, Size: 6
    at java.base/java.util.ArrayList.rangeCheckForAdd(ArrayList.java:756)
    at java.base/java.util.ArrayList.add(ArrayList.java:481)
    at DynamicArray.ArrayListPractice.main(ArrayListPractice.java:127)

```


1,20

Add new value using index-

```

118
119 //
120 ArrayList<String> footerList = new ArrayList<String>(30); //vc=30, pc=0
121 footerList.add("contact us");//0
122 footerList.add("help");//1
123 footerList.add("delivery info");//2
124 footerList.add("Returns");//3
125 footerList.add("cart");//4
126 footerList.add("accounts");//5
127 footerList.add(0, "Naveen");//IndexOutOfBoundsException
128 System.out.println(footerList);

```



```
[Naveen, contact us, help, delivery info, Returns, cart, accounts]
```

paste [arrlist27](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist27 {
6
7     //Boolean type generics.
8     public static void main(String[] args) {
9
10        //add from existing or sequential index
11        ArrayList<Boolean> studentnames=new ArrayList<Boolean>();
12        studentnames.add(true);
13        studentnames.add(false);
14        //add null
15        //allowed
16        studentnames.add(null);
17        studentnames.add(null);
18        studentnames.add(0,false); //overwrites first value.
19        studentnames.add(5,true);
20
21
22        System.out.println(studentnames);
23
24    }
25
26 }
27
28 //[false, true, false, null, null, true]
29

```

Set-

To update.

```

129
130         footerList.set(0, "Himani");
131         System.out.println(footerList);
132
133
134 [null, monika, monika, sunil, vidha, surya, sunil, null,
135 [Himani, help, delivery info, Returns, cart, accounts]
136

```

paste arrlist28

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist28 {
6
7     //Boolean type generics.
8     public static void main(String[] args) {
9
10        //use set to update values.
11        //add index also does update.
12        //for naveen it was not.
13
14        ArrayList<Boolean> studentnames=new ArrayList<Boolean>();
15        studentnames.add(true);
16        studentnames.add(false);
17        //add null
18        //allowed
19        studentnames.add(null);
20        studentnames.add(null);
21        studentnames.add(0,false); //overwrites first value.
22        studentnames.add(5,true);
23        studentnames.set(1, false);
24
25
26        System.out.println(studentnames);
27
28    }
29
30 }

```

```

29
30 }
31
32 //[false, false, false, null, null, true]
33

```

No access modifiers for collections.

We can add "" or " " in array list.

```

110 studentList.add(" "); //4
111 studentList.add(null);

```

paste arrlist29

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist29 {
6
7     //Boolean type generics.
8     public static void main(String[] args) {
9
10
11         ArrayList<Boolean> studentnames=new ArrayList<Boolean>();
12         studentnames.add(true);
13         studentnames.add(false);
14         //add null
15         //allowed
16         studentnames.add(null);
17         studentnames.add(null);
18         studentnames.add(0,false); //overwrites first value.
19         studentnames.add(5,true);
20         studentnames.set(1, false);
21         studentnames.add(" ");
22         //The method add(Boolean) in the type ArrayList<Boolean>
23         //is not applicable for the arguments (String)
24         studentnames.add("");
25         //The method add(Boolean) in the type ArrayList<Boolean>
26         //is not applicable for the arguments (String)
27

```

```

25         //The method add(Boolean) in the type ArrayList<Boolean>
26         //is not applicable for the arguments (String)
27
28
29
30         System.out.println(studentnames);
31
32     }
33
34 }
35

```

```

101 //
102 ArrayList<String> studentList = new ArrayList<String>();//vc=10, pc=0
103
104 studentList.add(null);//0
105 studentList.add("monika");//1
106 studentList.add("monika");//2
107 studentList.add("sunil");//3
108 studentList.add("vibha");//3
109 studentList.add("surya");//3
110 studentList.add("I");//4
111 studentList.add(null);
112 studentList.add(null);
113
114 System.out.println(studentList);
115

```



```
[null, monika, monika, sunil, vibha, surya, , null, null]
```

paste arrlist30

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist30 {
6
7     //String type generics.
8     public static void main(String[] args) {
9
10         ArrayList<String> studentnames=new ArrayList<String>();
11         studentnames.add("james");
12         studentnames.add("vibha");
13         studentnames.add("bond");
14         studentnames.add("james");
15         //add null
16         //allowed
17         studentnames.add(null);
18         studentnames.add(null);
19         //add space and blank
20         studentnames.add(" ");
21         studentnames.add("");
22
23         System.out.println(studentnames);
24
25     }
26 }

```

```

24
25     }
26
27 }
28
29 //[james, vibha, bond, james, null, null, , ]
30

```

```

115
116         System.out.println(studentList.get(1));
117

```

monika

```

58 //ArrayList with generics
59
60 ArrayList<Integer> numList = new ArrayList<Integer>();//vc=10, pc=0
61 numList.add(100);
62 numList.add(200);
63 numList.add(null);
64
65 for(Integer e : numList) {
66     System.out.println(e);
67 }
68

```

100

200

Null

To get values we can use primitive types where applicable but not good practice-

```

58 //ArrayList with generics
59
60 ArrayList<Integer> numList = new ArrayList<Integer>();//vc=10, pc=0
61 numList.add(100);
62 numList.add(200);
63 numList.add(null);
64
65 for(int e : numList) {
66     System.out.println(e);
67 }
68

```



paste [arrlist33](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist33 {
6
7     //integer type generics.
8     public static void main(String[] args) {
9
10        //null allowed in integer also.
11        ArrayList<Integer> studentnames=new ArrayList<Integer>();
12        studentnames.add(10);
13        studentnames.add(15);
14        studentnames.add(20);
15        studentnames.add(30);
16        //add null
17        //allowed
18        studentnames.add(null);
19        studentnames.add(null);
20
21        //To get values we can use primitive types
22        //where applicable but not good practice.
23        //we get errors when value is null.
24        //null is not primitive.
25        for(int e:studentnames) {
26            System.out.println(e);
27        }
28    }
29 }

```

```

28    }
29 }
30
31 }
32
33 //10
34 //15
35 //20
36 //30
37 //Exception in thread "main" java.lang.NullPointerException:
38 //  Cannot invoke "java.lang.Integer.intValue()"
39 //  because the return value of "java.util.Iterator.next()" is null
40 //  at com.day28.arrlist33.main(arrlist33.java:23)
41

```

[paste arrlist34](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist34 {
6
7     //integer type generics.
8     public static void main(String[] args) {
9
10        ArrayList<Integer> studentnames=new ArrayList<Integer>();
11        studentnames.add(10);
12        studentnames.add(15);
13        studentnames.add(20);
14        studentnames.add(30);
15
16        //cannot print null values when using int.
17        for(int e:studentnames) {
18            System.out.println(e);
19        }
20
21    }
22
23 }
24
25 //10
26 //15
27 //20
28 //30
29

```

~~1,37~~

Wrapper classes can contain only the data which it can hold. (1.42)

Example in primitive double can hold int also (lower values).

In wrapper, Double cannot hold int, compile error.

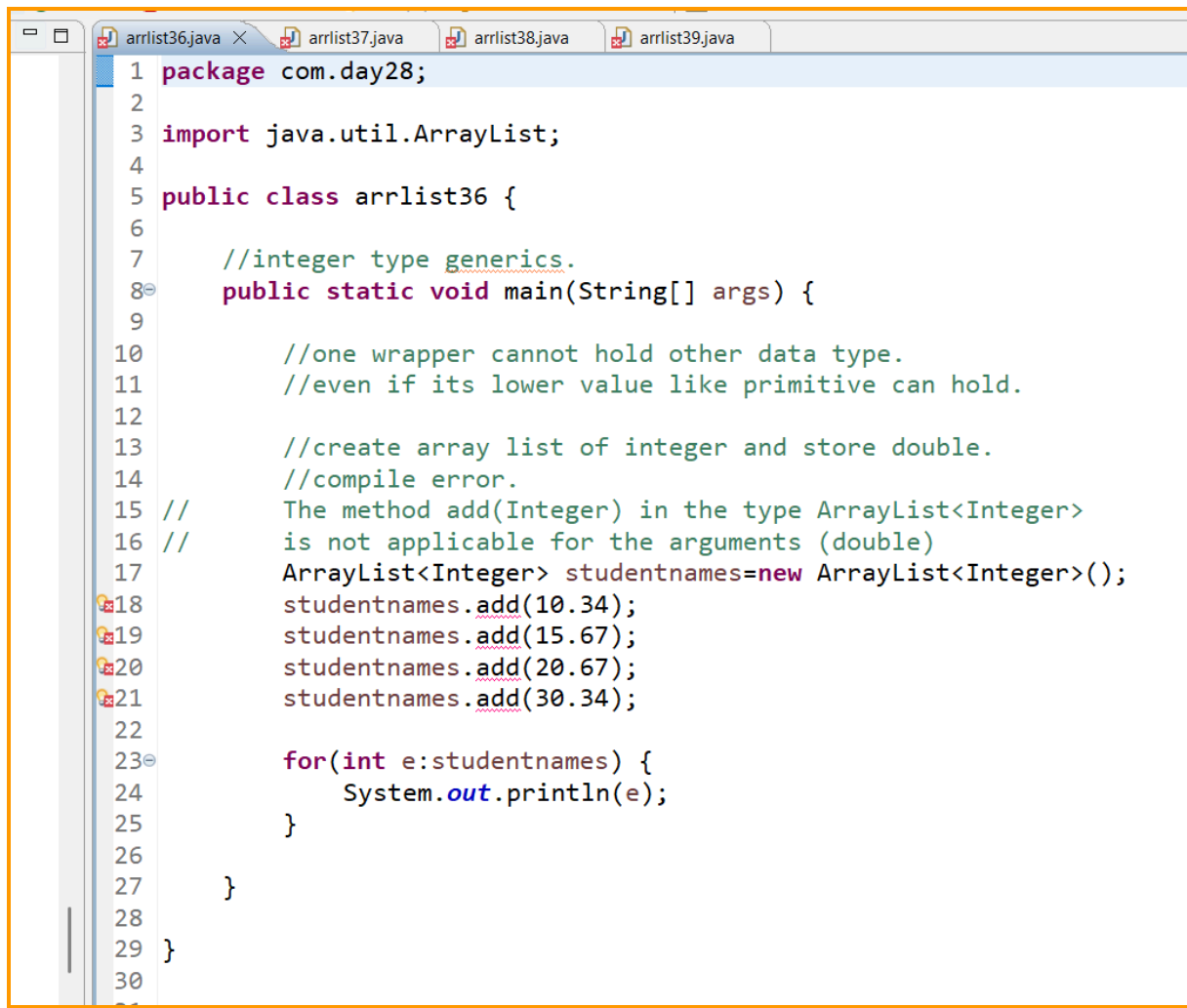
paste [arrlist35](#)


```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist35 {
6
7     //double type generics.
8     public static void main(String[] args) {
9
10        //one wrapper cannot hold other data type.
11        //even if its lower value like primitive can hold.
12
13        //create array list of double and store int.
14        //compile error.
15        //The method add(int, Double) in the type ArrayList<Double>
16        //is not applicable for the arguments (int)
17        ArrayList<Double> studentnames=new ArrayList<Double>();
18        studentnames.add(10);
19        studentnames.add(15);
20        studentnames.add(20);
21        studentnames.add(30);
22
23        for(double e:studentnames) {
24            System.out.println(e);
25        }
26    }
27 }
28
29 }
30

```

[paste arrlist36](#)



The screenshot shows an IDE window with four tabs: arrlist36.java, arrlist37.java, arrlist38.java, and arrlist39.java. The active tab is arrlist36.java, which contains the following Java code:

```
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist36 {
6
7     //integer type generics.
8     public static void main(String[] args) {
9
10         //one wrapper cannot hold other data type.
11         //even if its lower value like primitive can hold.
12
13         //create array list of integer and store double.
14         //compile error.
15         // The method add(Integer) in the type ArrayList<Integer>
16         // is not applicable for the arguments (double)
17         ArrayList<Integer> studentnames=new ArrayList<Integer>();
18         studentnames.add(10.34);
19         studentnames.add(15.67);
20         studentnames.add(20.67);
21         studentnames.add(30.34);
22
23         for(int e:studentnames) {
24             System.out.println(e);
25         }
26     }
27 }
28
29 }
30
```

There are four compilation errors indicated by red squiggly lines and error icons on the left margin:

- Line 18: `studentnames.add(10.34);` - Error: The method `add(Integer)` in the type `ArrayList<Integer>` is not applicable for the arguments `(double)`.
- Line 19: `studentnames.add(15.67);` - Error: The method `add(Integer)` in the type `ArrayList<Integer>` is not applicable for the arguments `(double)`.
- Line 20: `studentnames.add(20.67);` - Error: The method `add(Integer)` in the type `ArrayList<Integer>` is not applicable for the arguments `(double)`.
- Line 21: `studentnames.add(30.34);` - Error: The method `add(Integer)` in the type `ArrayList<Integer>` is not applicable for the arguments `(double)`.

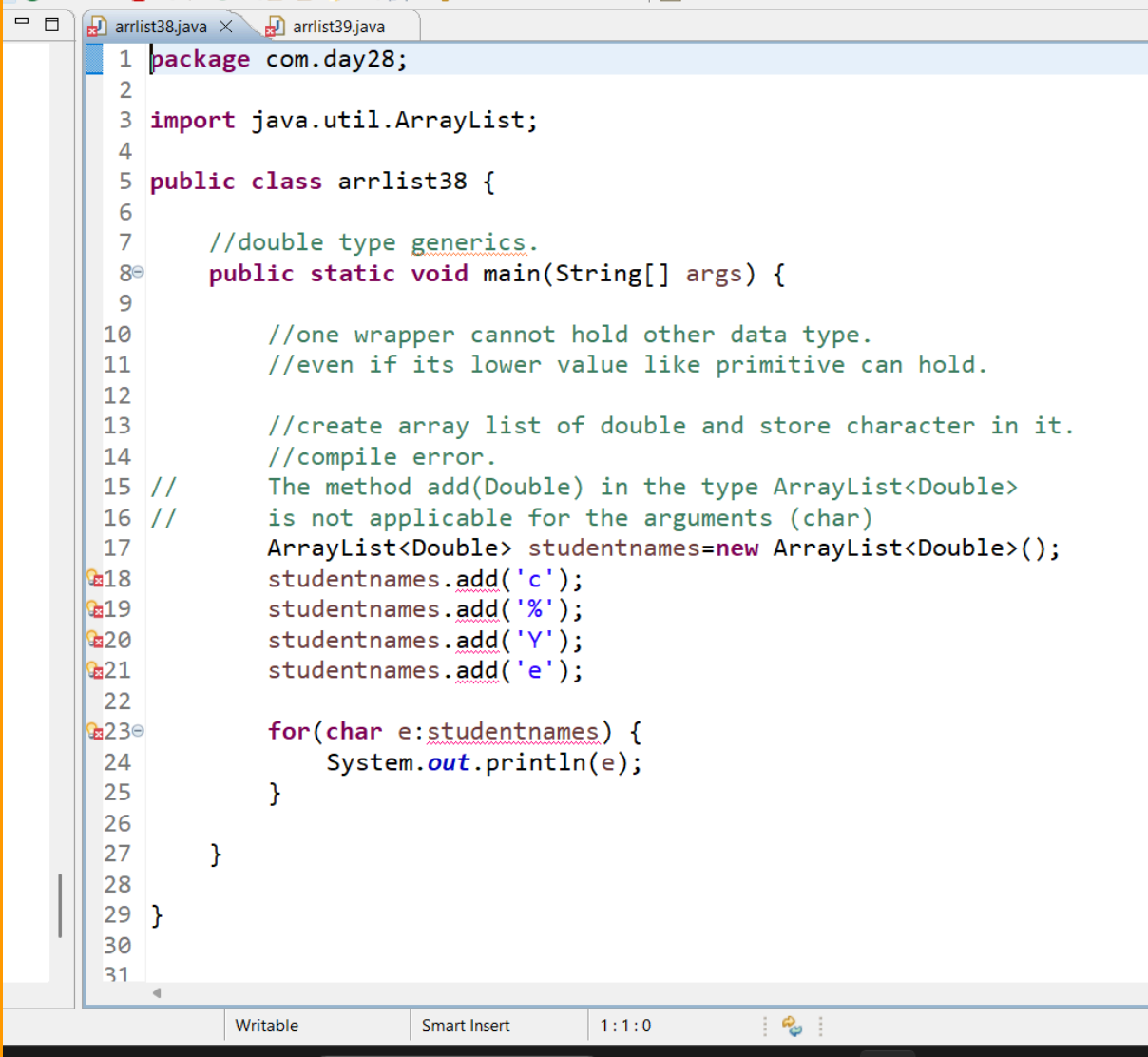
[paste arrlist37](#)

```

1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist37 {
6
7     //double type generics.
8     public static void main(String[] args) {
9
10        //one wrapper cannot hold other data type.
11        //even if its lower value like primitive can hold.
12
13        //create array list of double and store float in it.
14        //compile error.
15        // The method add(Double) in
16        // the type ArrayList<Double> is
17        // not applicable for the arguments (float)
18        ArrayList<Double> studentnames=new ArrayList<Double>();
19        studentnames.add(10.34f);
20        studentnames.add(15.67f);
21        studentnames.add(20.67f);
22        studentnames.add(30.34f);
23
24        for(double e:studentnames) {
25            System.out.println(e);
26        }
27
28    }
29
30 }
31

```

[paste arrlist38](#)



```
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist38 {
6
7     //double type generics.
8     public static void main(String[] args) {
9
10        //one wrapper cannot hold other data type.
11        //even if its lower value like primitive can hold.
12
13        //create array list of double and store character in it.
14        //compile error.
15        // The method add(Double) in the type ArrayList<Double>
16        // is not applicable for the arguments (char)
17        ArrayList<Double> studentnames=new ArrayList<Double>();
18        studentnames.add('c');
19        studentnames.add('%');
20        studentnames.add('Y');
21        studentnames.add('e');
22
23        for(char e:studentnames) {
24            System.out.println(e);
25        }
26    }
27 }
28
29 }
30
31
```

[paste arrlist39](#)

```
arrlist39.java ×
1 package com.day28;
2
3 import java.util.ArrayList;
4
5 public class arrlist39 {
6
7     //double type generics.
8     public static void main(String[] args) {
9
10        //one wrapper cannot hold other data type.
11        //even if its lower value like primitive can hold.
12
13        //create array list of double and store character in it.
14        //compile error.
15        // The method add(Double) in the type ArrayList<Double>
16        // is not applicable for the arguments (String)
17        ArrayList<Double> studentnames=new ArrayList<Double>();
18        studentnames.add("tiger");
19        studentnames.add("lion");
20        studentnames.add("goat");
21        studentnames.add("cheetah");
22
23        for(char e:studentnames) {
24            System.out.println(e);
25        }
26
27    }
28
29 }
30
31
```