

Handwritten notes and diagrams illustrating string processing and substring generation.

Top left: A diagram showing a string "4931" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Top right: A diagram showing a string "499" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Center: A diagram showing a string "81615" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Bottom left: A diagram showing a string "81615" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Bottom right: A diagram showing a string "81615" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

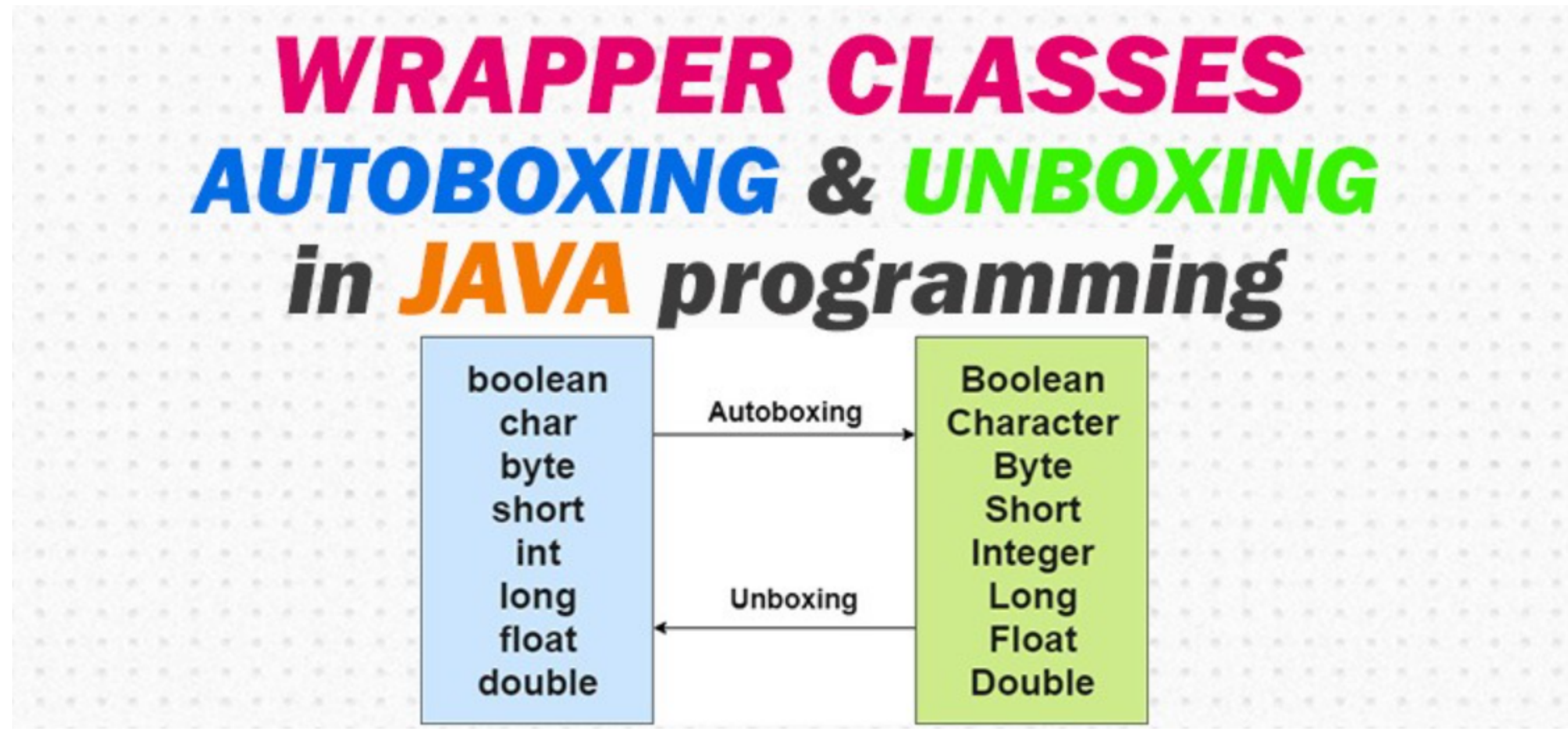
```
public static void printallSubString(String s) {
    int c = 0;
    boolean[] visited = new boolean[s.length()];
    for (int len = 1; len <= s.length(); len++) {
        for (int j = len; j <= s.length(); j++) {
            int i = j - len;
            String s1 = s.substring(i, j); // char i to j-1
            if (IsNumber(Long.parseLong(s1)) == true && isvisited(visited, i, j) == true) {
                c++;
                for (int k = i; k < j; k++) { // marked kra
                    visited[k] = true;
                }
            }
        }
    }
    System.out.println(c);
}
```

Handwritten notes and diagrams illustrating the logic of the code, including a diagram showing a string "127" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Primitive data type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    int a = 10;
    Integer a1 = 12;
    System.out.println(a);
    System.out.println(a1);
}
```

Handwritten notes and diagrams illustrating the concept of memory and data storage, including a diagram showing a box labeled "a=10" and "a1=20" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.



Handwritten notes and diagrams illustrating the concept of autoboxing and unboxing, including a diagram showing a box labeled "Integer c1=71;" and "Integer c2=71;" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Handwritten notes and diagrams illustrating the concept of autoboxing and unboxing, including a diagram showing a box labeled "Integer c1=71;" and "Integer c2=71;" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    ArrayList<Integer> ll = new ArrayList<>();
    ll.add(10); // 0(1)
    ll.add(20);
    ll.add(30);
    ll.add(120);
    ll.add(130);
    ll.add(7); // 0(N)
    ll.add(2);
    ll.add(21);
    ll.add(12);
    ll.add(211);
    ll.add(3);
    System.out.println(ll);
}
```

Handwritten notes and diagrams illustrating the logic of the code, including a diagram showing a list of integers [10, 20, 30, 120, 130, 7, 2, 21, 12, 211, 3] with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Handwritten notes and diagrams illustrating the concept of array operations, including a diagram showing a box labeled "no of Operation" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

Handwritten notes and diagrams illustrating the concept of array operations, including a diagram showing a box labeled "Input" with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

```
public static void Sum_Of_Two_Arrays(int[] arr1, int[] arr2) {
    ArrayList<Integer> ll = new ArrayList<>();
    int i = arr1.length - 1, j = arr2.length - 1;
    int carry = 0;
    while (i >= 0 && j >= 0) {
        int sum = carry + arr1[i] + arr2[j];
        ll.add(sum % 10);
        carry = sum / 10;
        i--;
        j--;
    }
    while (i >= 0) {
        int sum = carry + arr1[i];
        ll.add(sum % 10);
        carry = sum / 10;
        i--;
    }
    while (j >= 0) {
        int sum = carry + arr2[j];
        ll.add(sum % 10);
        carry = sum / 10;
        j--;
    }
}
```

Handwritten notes and diagrams illustrating the logic of the code, including a diagram showing a list of integers [5, 1, 3, 8, 5, 4, 3] with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.

```
public static void Sum_Of_Two_Arrays(int[] arr1, int[] arr2) {
    ArrayList<Integer> ll = new ArrayList<>();
    int i = arr1.length - 1, j = arr2.length - 1;
    int carry = 0;
    while (i >= 0 && j >= 0) {
        int sum = carry + arr1[i] + arr2[j];
        ll.add(sum % 10);
        carry = sum / 10;
        i--;
        j--;
    }
    while (i >= 0) {
        int sum = carry + arr1[i];
        ll.add(sum % 10);
        carry = sum / 10;
        i--;
    }
    while (j >= 0) {
        int sum = carry + arr2[j];
        ll.add(sum % 10);
        carry = sum / 10;
        j--;
    }
}
```

Handwritten notes and diagrams illustrating the logic of the code, including a diagram showing a list of integers [5, 1, 3, 8, 5, 4, 3] with arrows indicating a path through its digits, possibly representing a sequence of operations or a search process.