ArrayDeque<E> – Java Stack Implementation

- Creating a Stack

```
ArrayDeque<Integer> stack = new ArrayDeque<>();
```

- Adding elements at the top of the stack

```
stack.push(element);
```

- Removing elements

```
Integer element = stack.pop();
```

- Getting the value of the topmost element

```
Integer element = stack.peek();
```

Stack – Utility Methods

```
ArrayDeque<Integer> stack = new ArrayDeque<>();

int size = stack.size();
boolean isEmpty = stack.isEmpty();
boolean exists = stack.contains(2);
```

Problem: Decimal to Binary Converter

- Create a converter which takes a **decimal number** and **converts it into a binary number**

Input
10
1024

Output
1010
10000000000

Check your solution here: https://judge.softuni.bg/Contests/1437/Stacks-and-Queues-Lab

Solution: Decimal to Binary Converter

```java
Scanner scanner = new Scanner(System.in);
int decimal = Integer.valueOf(scanner.nextLine());

ArrayDeque<Integer> stack = new ArrayDeque<>();

// TODO: check if number is 0

while (decimal != 0)
  stack.push(decimal % 2);
  decimal /= 2;

while (!stack.isEmpty())
  System.out.print(stack.pop());
```

Check your solution here: https://judge.softuni.bg/Contests/1437/Stacks-and-Queues-Lab

Problem: Matching Brackets

- We are given an arithmetical expression with brackets (with nesting)
- Goal: extract all sub-expressions in brackets

```
1 + (2 - (2 + 3) * 4 / (3 + 1)) * 5
```

```
(2 + 3)
(3 + 1)
(2 - (2 + 3) * 4 / (3 + 1))
```

Check your solution here: https://judge.softuni.bg/Contests/1437/Stacks-and-Queues-Lab

Solution: Matching Brackets (1)

```java
Scanner scanner = new Scanner(System.in);
String expression = scanner.nextLine();

Deque<Integer> stack = new ArrayDeque<>();

// continue…
```

Check your solution here: https://judge.softuni.bg/Contests/1437/Stacks-and-Queues-Lab

Solution: Matching Brackets (2)

```
for (int i = 0; i < expression.length(); i++) {
  char ch = expression.charAt(i);
  if (ch == '(')
    stack.push(i);
  else if (ch == ')')
    int startIndex = stack.pop();
    String contents =
    expression.substring(startIndex, i + 1);
    System.out.println(contents);
}
```

Check your solution here: https://judge.softuni.bg/Contests/1437/Stacks-and-Queues-Lab

ArrayDeque<E> – Java Queue Implementation (1)

- Creating a Queue

```
ArrayDeque<Integer> queue = new ArrayDeque<>();
```

- Adding elements at the end of the queue

```
queue.add(element);
queue.offer(element);
```

  - `add()` – throws exception if queue is full
  - `offer()` – returns false if a queue is full

ArrayDeque<E> – Java Queue Implementation (2)

- Removing elements

```java
element = queue.remove();
element = queue.poll();
```

  - `remove()` - throws exception if queue is empty
  - `poll()` - returns null if queue is empty
- Check first element

```java
element = queue.peek();
```

ArrayDeque<E> – Java Queue Implementation (3)

- Utility Methods

```
Integer element = queue.peek();
Integer size = queue.size();
Integer[] arr = queue.toArray();
boolean exists = queue.contains(element);
```

- `peek()` - checks the value of the first element
- `size()` - returns queue size
- `toArray()` - converts the queue to an array
- `contains()` - checks if element is in the queue