Stack and Queue

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

1. Algorithmic Complexity
2. Stack - Last In First Out (LIFO)
   - Stack Functionality
   - Java Stack Implementation
   - Overview of All Operations
3. Queue - First In First Out(FIFO)
   - Queue Functionality
   - Java Stack Implementation
   - Overview of All Operations
4. Priority Queue

Have a Question?

Software University

[sli.do](sli.do)

#java-advanced

Algorithmic Complexity

Algorithmic Complexity

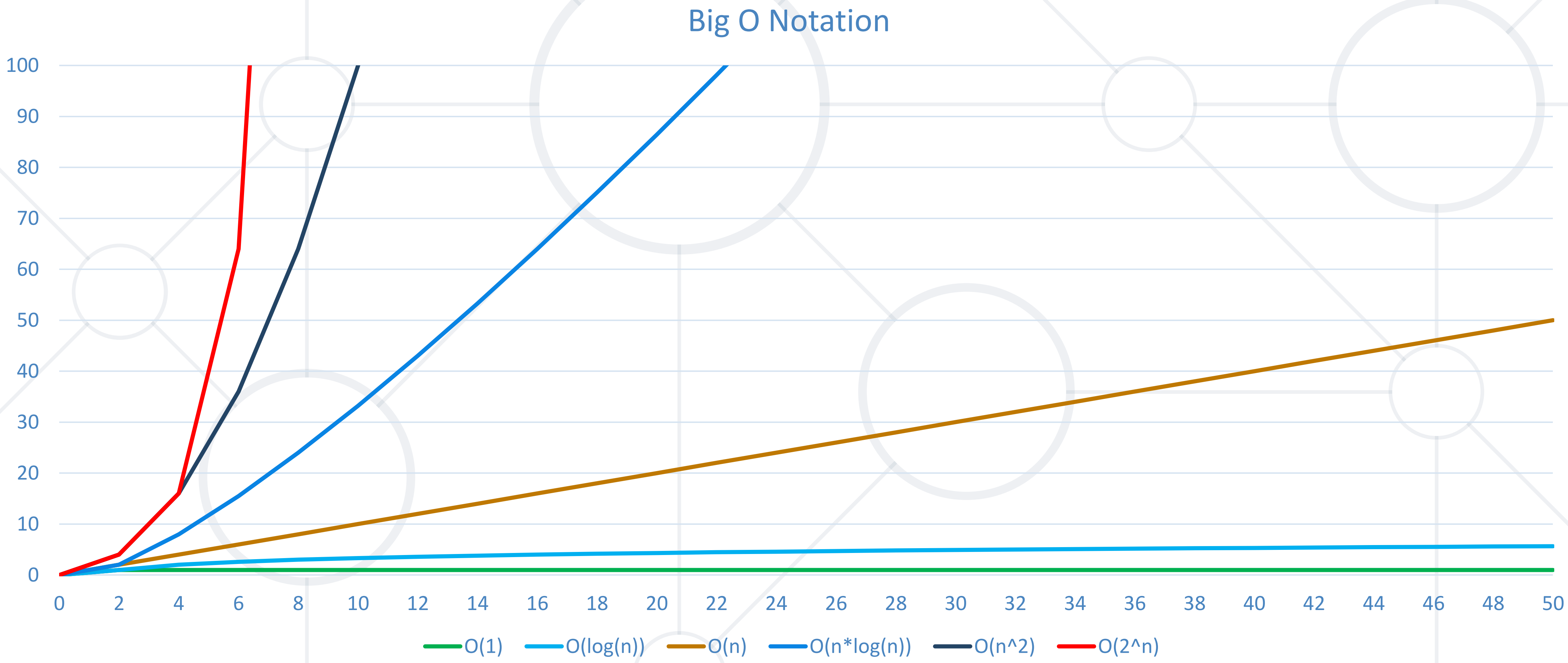
- Describes performance of particular algorithm
  - Runtime and memory consumption based on the input size **N**
  - We usually care about the **worst-case** performance
- We measure the complexity as the **Big O notation**
  - Numerical function depending on the input size **O(N)**
  - We measure time as the number of **simple steps**
  - We measure memory as input data **N** by it's **type size**

Algorithmic Complexity

- O(1) – Constant time – time does not depend on **N**
- O(log(N)) – Logarithmic time – grows with rate as **log(N)**
- O(N) – Linear time grows at the same rate as **N**
- O(N^2),O(N^3) – Quadratic, Cubic grows as square or cube of **N**
- O(2^N) – Exponential grows as **N** becomes the exponent worst algorithmic complexity
  - For input size of 10 - 1024 steps
  - For input size of 100 – 1267650600228229401496703205376 steps
- http://bigocheatsheet.com/

Asymptotic Functions

- Below are some examples of **common algorithmic** grow:



Get Sum Number of Steps

- Calculate maximum steps to find sum of even elements in an array

```
int getSumEven(int[] array) {
  int sum = 0;
  for (int i = 0; i < array.length; i++)
    if (array[i] % 2 == 0) sum += array[i];
  return sum;
}
```

Solution:

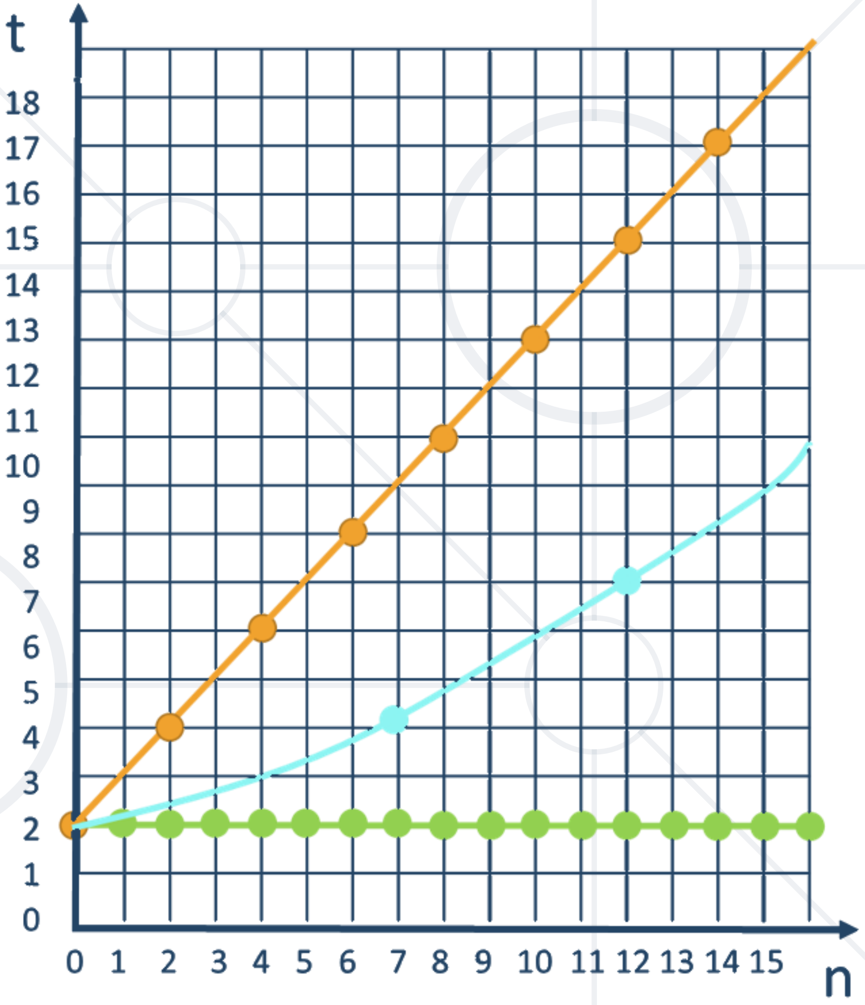
T(n) = 9n + 3

Counting maximum steps is called **worst-case** analysis

- Assume that a **single step** is a single CPU instruction:
  - assignments, array lookups, comparisons, arithmetic operations

Time Complexity

- **Worst-case**
  - An upper bound on the running time
- **Average-case**
  - Average running time
- **Best-case**
  - The lower bound on the running time (the optimal case)



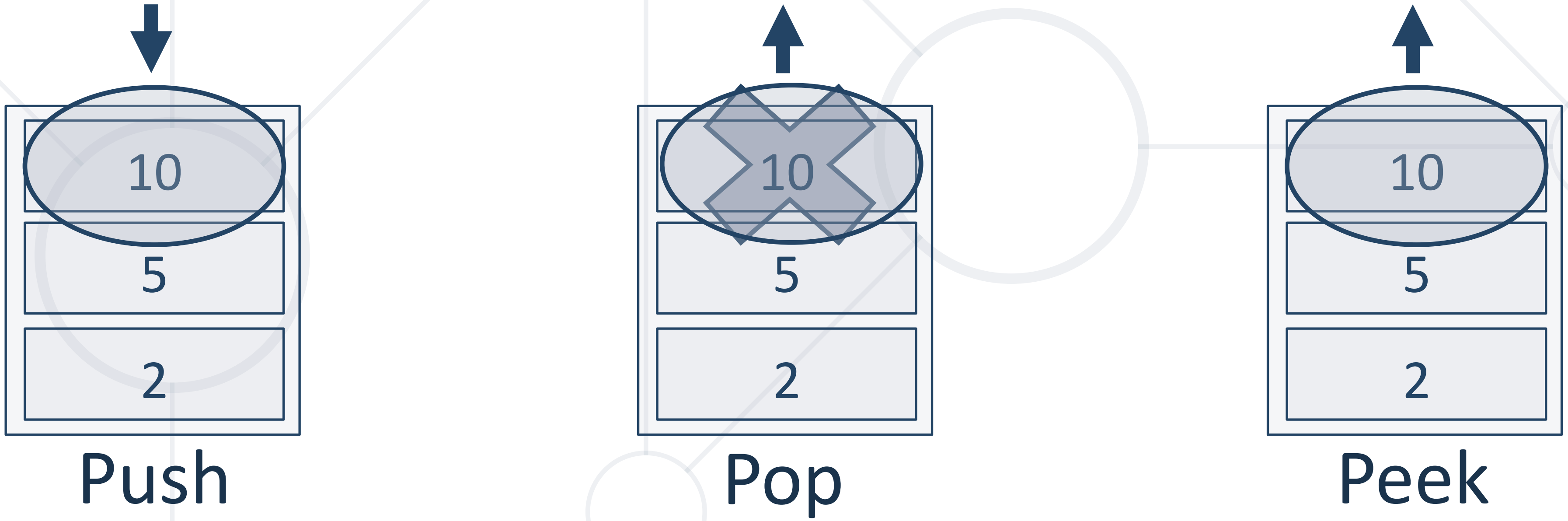
Stacks and Queue vs. ArrayDeque

- Why don't use Stack and Queue?
  - Implementation details which make **unsecure usability**
  - In many cases those structures will **decrease the performance**
- Why to use ArrayDeque?
  - Implementation which makes the structure **more secure**
  - **Better performance** and usability
  - Methods which operate as those structures suggest

Stack

Stack Functionality

- Stacks provide the following functionality:
  - Pushing an element at the top of the stack
  - Popping element from the top of the stack
  - Getting the topmost element without removing it



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);
```

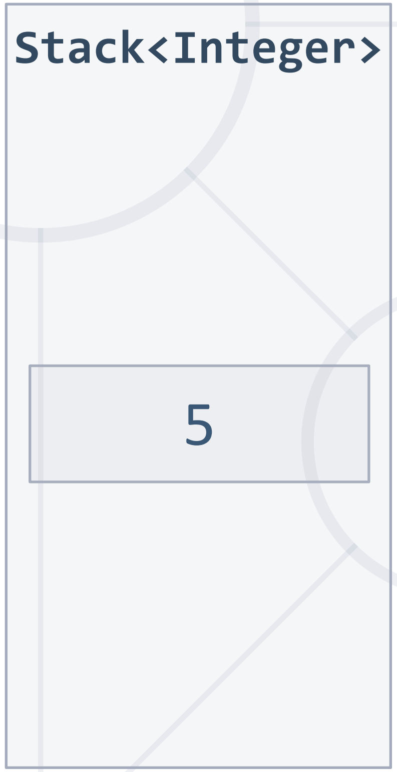
Stack – Overview of All Operations

132

`push()`

`pop()`

`peek()`



Stack<Integer>

5

`size():` 5

Problem: Browser History

- Write a program which takes 2 types of **browser instructions**:
  - Normal navigation: a **URL** is set, given by a string
  - The string **"back"** that sets the current URL to the last set URL

Input

```
https//softuni.bg/
back
https//softuni.bg/trainings/courses
back
https//softuni.bg/trainings/2056
back
https//softuni.bg/trainings/live
https//softuni.bg/trainings/live/details
Home
```

Output

```
https//softuni.bg/
no previous URLs
https//softuni.bg/trainings/courses
https//softuni.bg/
https//softuni.bg/trainings/2056
https//softuni.bg/
https//softuni.bg/trainings/live
https//softuni.bg/trainings/live/details
```

Solution: Browser History (1)

```
Scanner scanner = new Scanner(System.in);

ArrayDeque<String> browser = new ArrayDeque<>();
String line = scanner.nextLine();

String current = "";

// continue…
```

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

Solution: Browser History (2)

```
while(!line.equals("Home")) {
  if(line.equals("back")) {
    if(!browser.isEmpty()) { current = browser.pop();
    } else {
    System.out.println("no previous URLs");
    line = scanner.nextLine();
    continue; }
  } else {
    if(!current.equals("")) { browser.push(current); }
    current = line; }
  System.out.println(current);
  line = scanner.nextLine(); }
```

Problem: Simple Calculator

- Implement a simple calculator that can evaluate simple expressions (only addition and subtraction)

Input
`2 + 5 + 10 - 2 - 1`
`2 - 2 + 5`

Output
14
5

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

Solution: Simple Calculator (1)

```
Scanner scanner = new Scanner(System.in);
String[] tokens = scanner.nextLine().split("\\s+");

Deque<String> stack = new ArrayDeque<>();
Collections.addAll(stack, tokens);

// continues…
```

Split by regex

Adds a collection to another collection

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

Solution: Simple Calculator (2)

```
while (stack.size() > 1) {
  int first = Integer.valueOf(stack.pop());
  String op = stack.pop();
  int second = Integer.valueOf(stack.pop());

  switch (op)
    case "+": stack.push(String.valueOf(first + second));
      break;
    case "-": stack.push(String.valueOf(first - second));
      break;
}

System.out.println(stack.pop());
```

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

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

```
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)

```
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)

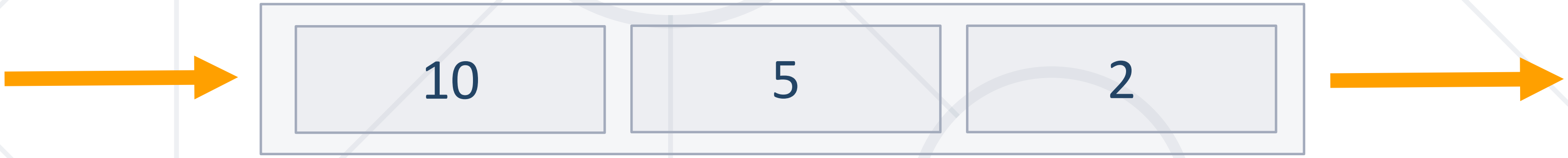
```
for (int i = 0; i < expression.length(); i++) {
  char ch = expression.charAt(i);
  if (ch == '(')
    stack.push(i);
  else if (ch == ')')
    int startIndex = stack.pop();
    String contents =
    expression.substring(startIndex, i + 1);
    System.out.println(contents);
}
```

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

Queues

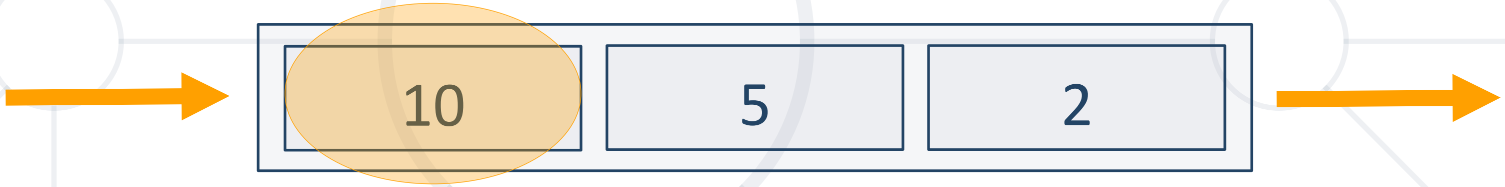
Queue

- **First In First Out**

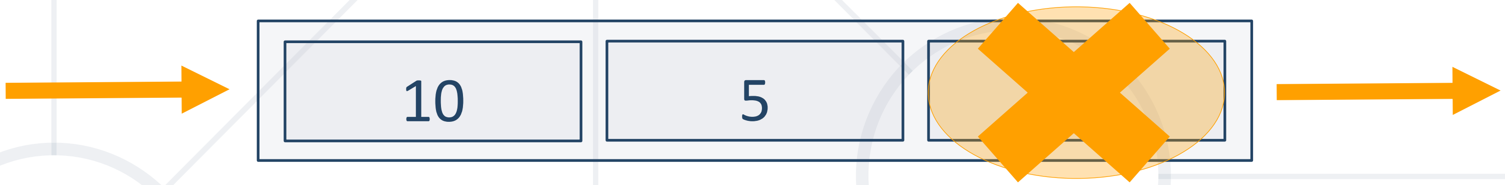


Queue – Abstract Data Type

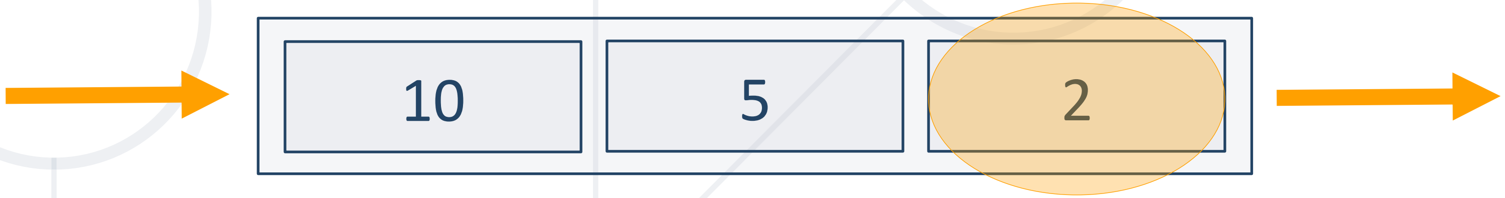
- Queues provide the following functionality:
  - Adding an element at the end of the queue



  - Removing the first element from the queue



  - Getting the first element of the queue without removing it



ArrayDeque<E> – Java Queue Implementation

- 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 queue is full

ArrayDeque<E> – Java Queue Implementation (2)

- Removing elements

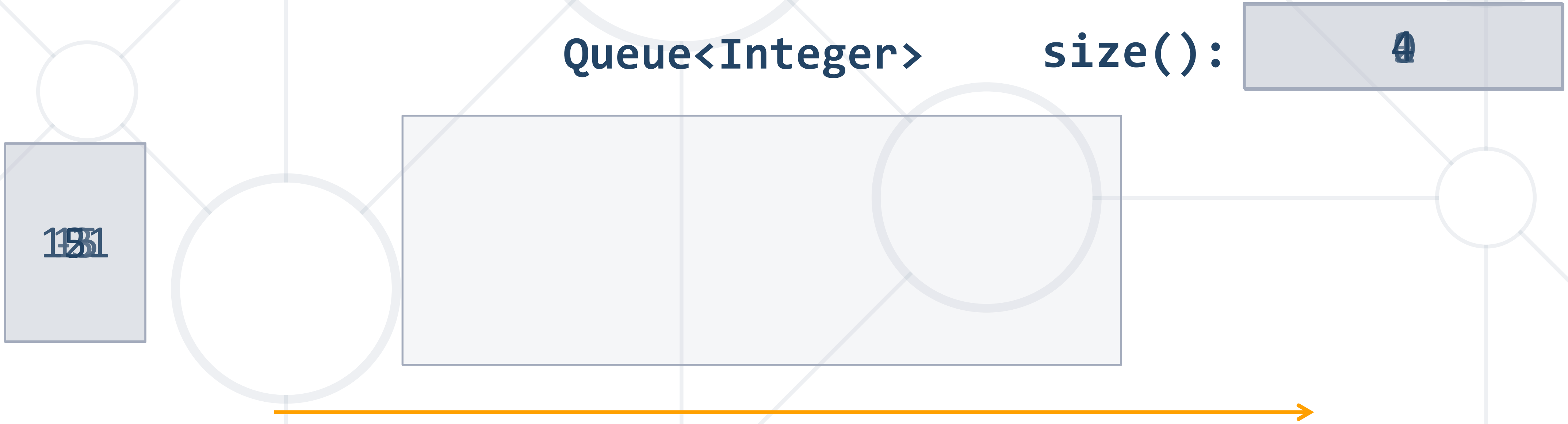
```java
element = queue.remove();
element = queue.poll();
```

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

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

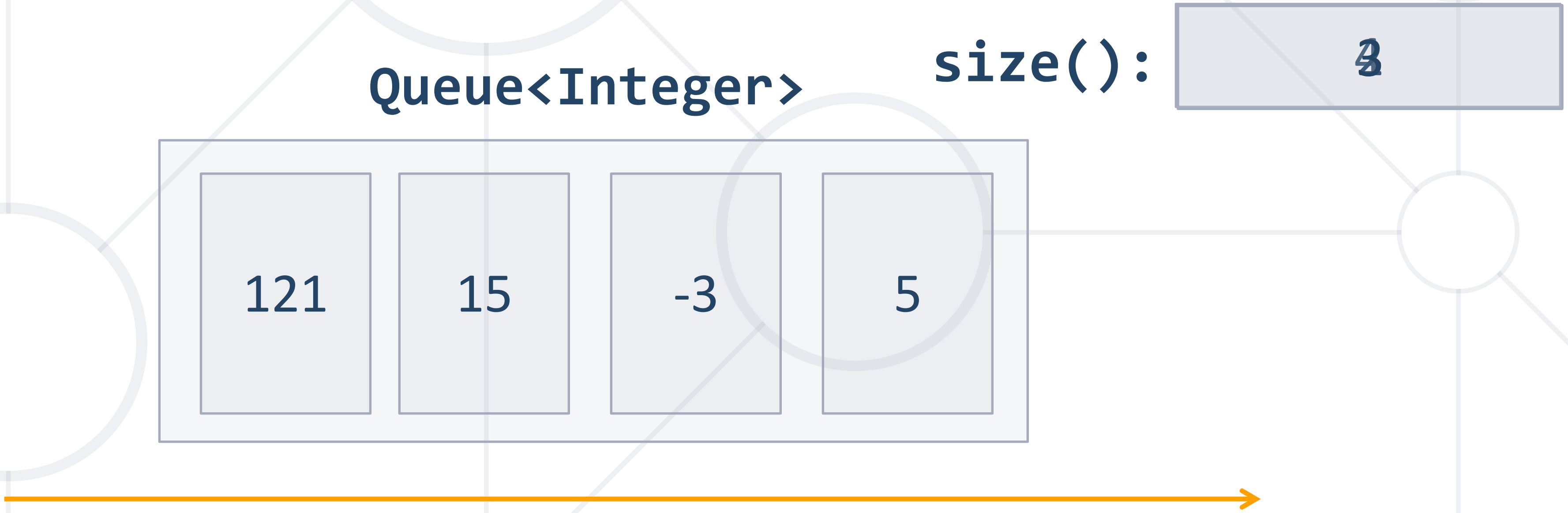
Add() / Offer()

- Adds an element to the queue



Remove() / Poll()

- Returns and removes first element



Problem: Hot Potato

- Children form a circle and pass a hot potato clockwise
- Every nth toss a child is removed until only one remains
- Upon removal the potato is passed forward
- Print the child that remains last

Input	Output

`Sam John Sara`
`2`
	Removed John
	Removed Sam
	Last is Sara

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

Solution: Hot Potato (1)

```
Scanner scanner = new Scanner(System.in);
String[] children = scanner.nextLine().split("\\s+");
int n = Integer.valueOf(scanner.nextLine());

ArrayDeque<String> queue = new ArrayDeque<>();

for (String child : children)
  queue.offer(child);

// continue…
```

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

Solution: Hot Potato (2)

```
while (queue.size() > 1) {
  for (int i = 1; i < n; i++)
    queue.offer(queue.poll());

  System.out.println("Removed " + queue.poll());
}

System.out.println("Last is " + queue.poll());
```

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

ArrayDeque<E> – Java Queue Implementation (3)

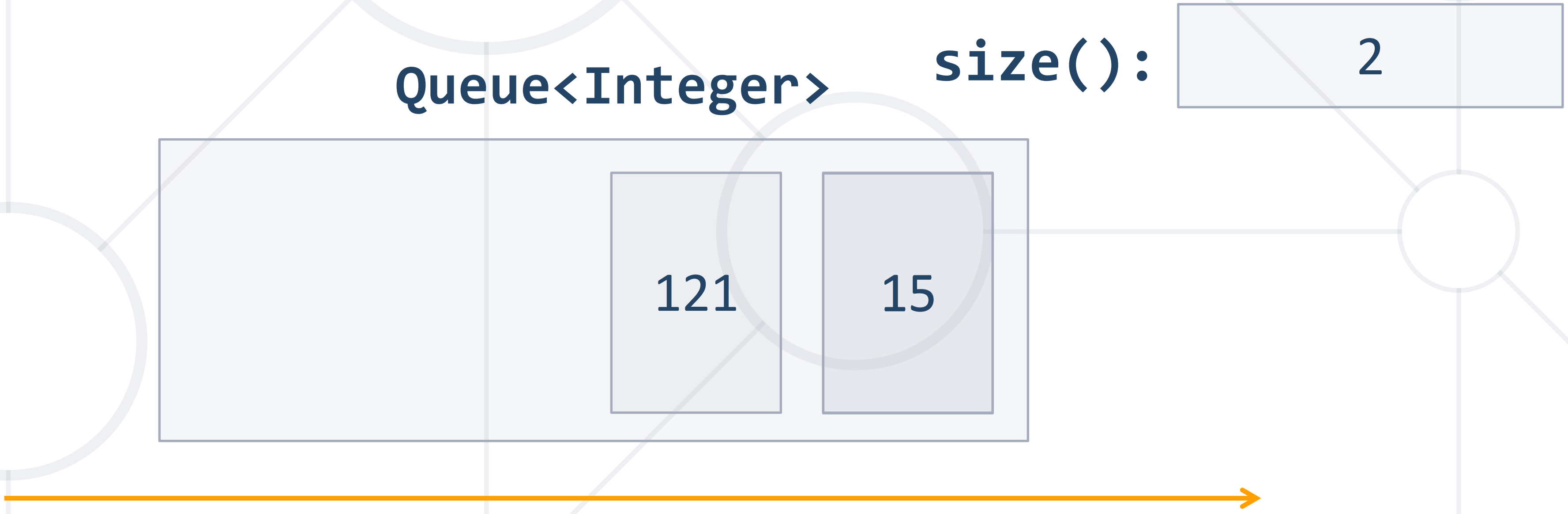
- Utility Methods

```
Integer element = queue.peeк();
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

Peek()

- Gets the first element without removing it



Problem: Math Potato

- Rework the previous problem so that a child is removed only on a prime cycle (cycles start from 1)
- If a cycle is not prime, just print the child's name

Input

```
Maria Peter George
2
```

→

Output

```
Removed Peter
Prime Maria
Prime George
Removed Maria
Last is George
```

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

Solution: Math Potato

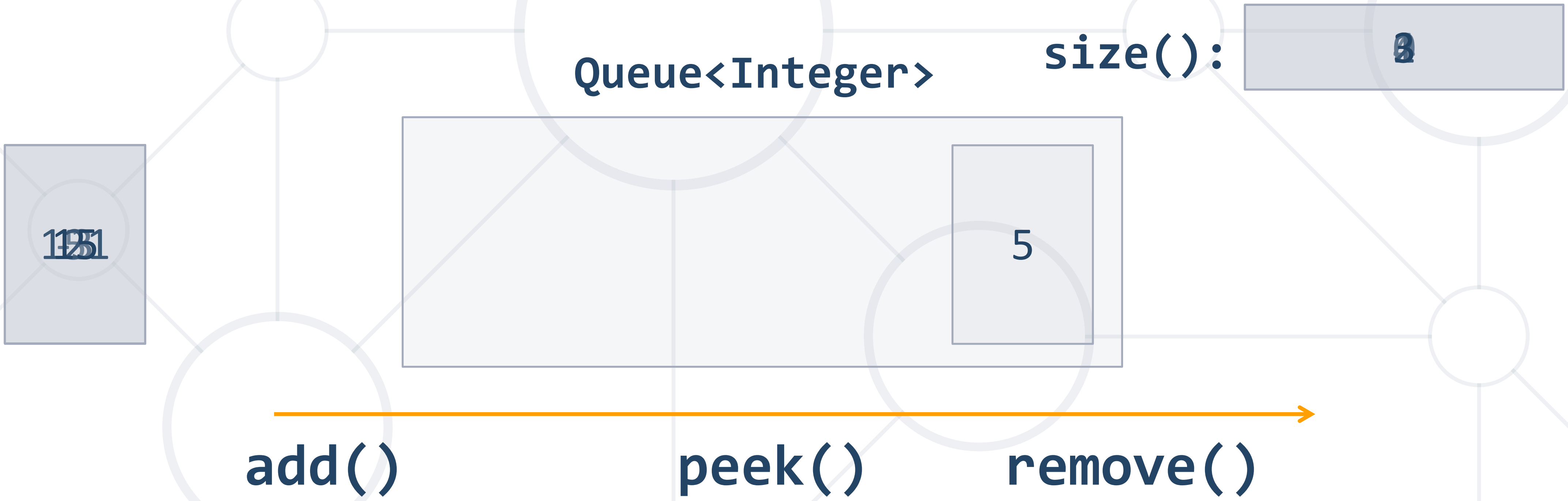
```
int cycle = 1;
while (queue.size() > 1) {
  for (int i = 1; i < n; i++)
    queue.offer(queue.poll());

  if (isPrime(cycle))
    System.out.println("Prime " + queue.peek());
  else
    System.out.println("Removed " + queue.poll());

  cycle++;
}
System.out.println("Last is " + queue.poll());
```

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

Queue – Overview of All Operations



Queue<Integer>

size(): 3

1251

5

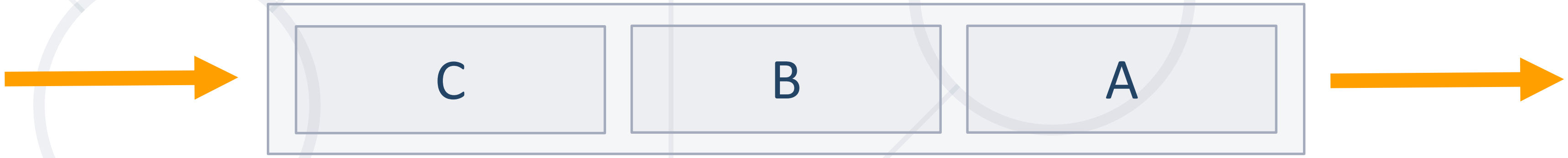
add()

peek()

remove()

Priority Queue

- Retains a specific order to the elements
- Higher priority elements are pushed to the beginning of the queue
- Lower priority elements are pushed to the end of the queue

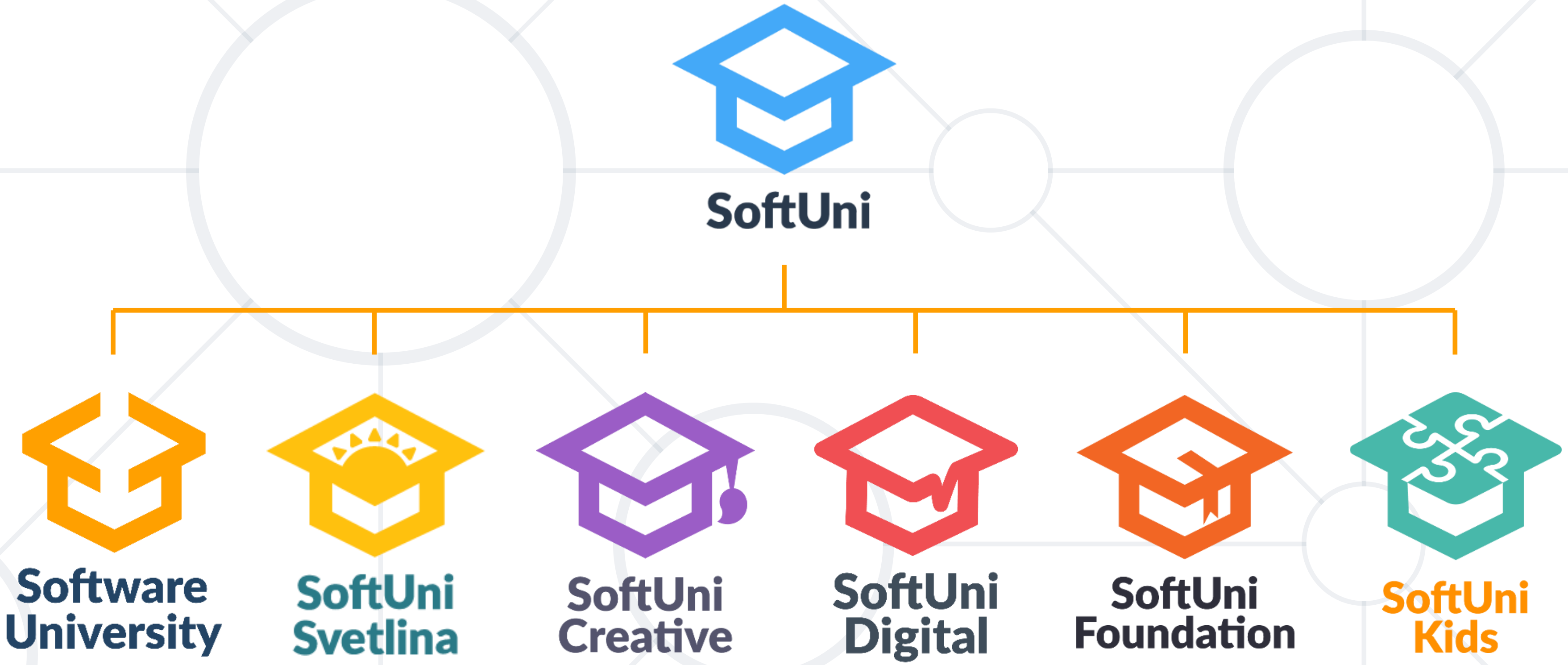


Summary

- Algorithmic Complexity
- **Stack** - Last In First Out (LIFO)
  - push(), pop(), peek()
- **Queue** - First In First Out (FIFO)
  - add(), poll(), peek()
- **Priority Queue**

Software University

Questions?



SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

SmartIT

XS SOFTWARE

zühlke
empowering ideas

INFRAGISTICS®

Coca-Cola HBC Bulgaria

DRAFT KINGS

Postbank
Решения за твоето утре

SOFTWARE GROUP

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg
- © Software University – https://softuni.bg