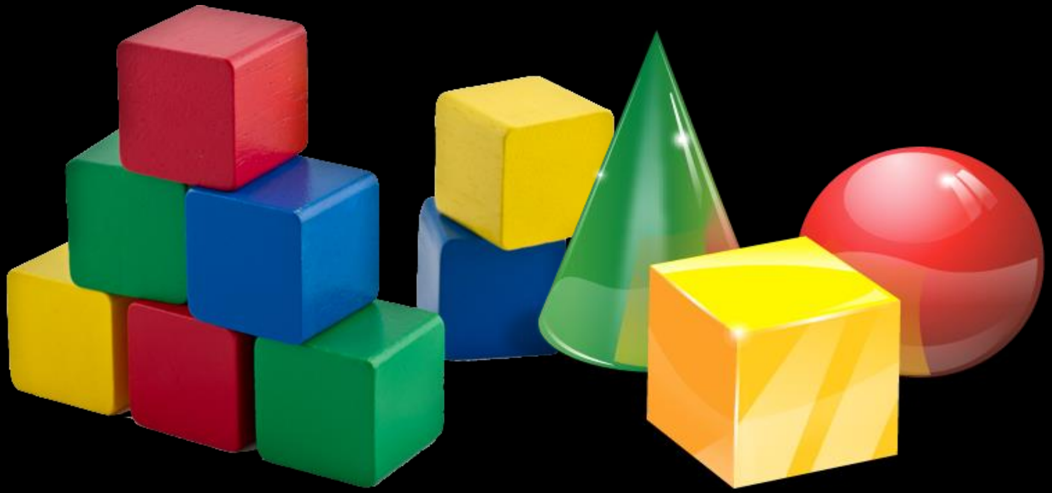
Objects, Classes and Collections

Using Objects and Classes
Processing Sequences of Elements



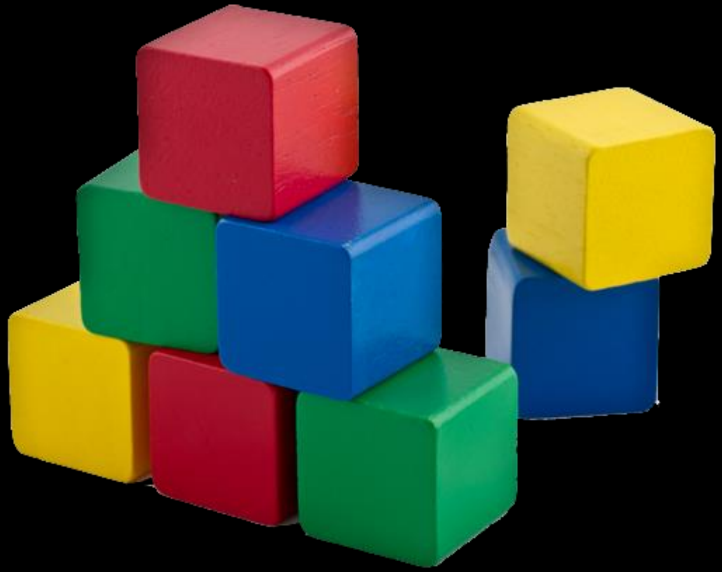
SoftUni Team
Technical Trainers
Software University
http://softuni.bg

Table of Contents

Questions

sli.do

#JavaFundamentals



Objects and Classes

What is an Object? What is a Class?

Classes

- Classes provide **structure for describing and creating objects**
  - Act as **template** for objects of the **same type**

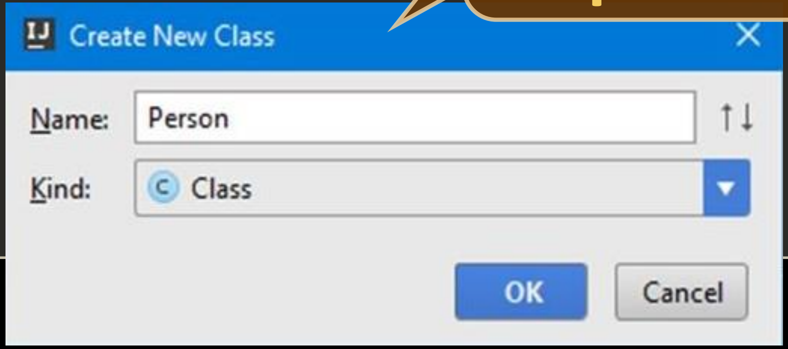
Keyword

Class **name**

```
class Person {
   …
}
```

Class **body**

Class in a **separate file**



Name: Person

Kind: Class

Class Members

- Class is made up of **state** and behavior
- Fields **store state** (data)
- Methods **describe behaviour**

```
class Person {
  String name;
  String birthdate;
  String gender;

  int calculateAge(){ … }
}
```

Fields

Method

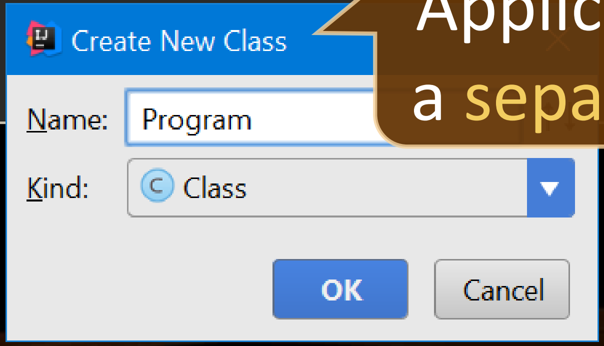
Creating an Object

- A class can have **many instances** (objects)

```
class Program {
  public static void main(String args) {
    Person Gosho = new Person();
    Person Mariika = new Person();
  }
}
```

Variable stores a **reference**

Use the **new** keyword



Name: Program

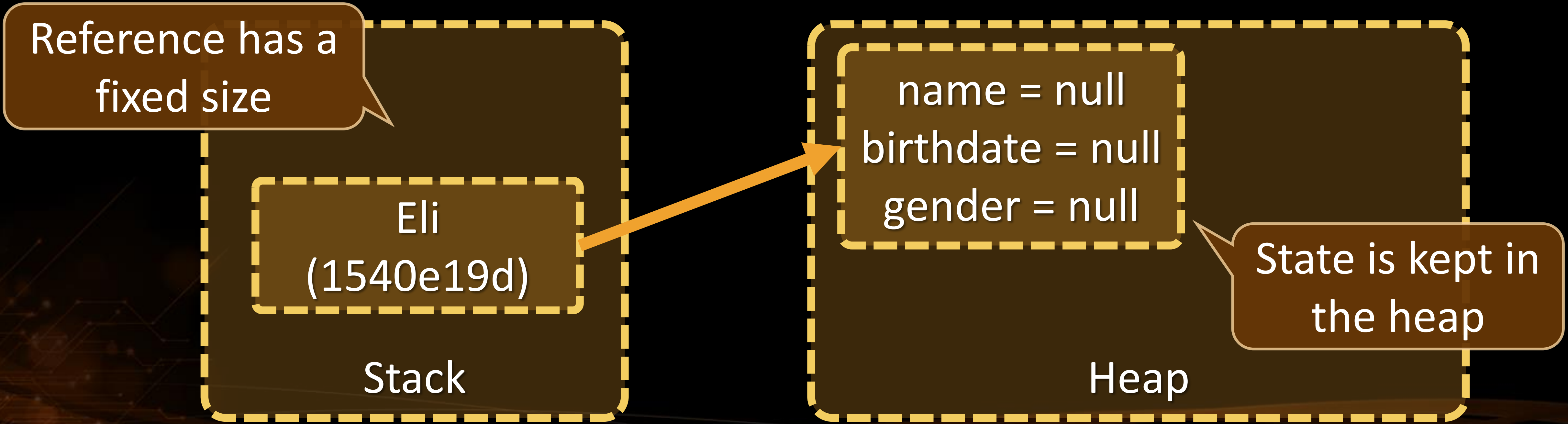
Kind: Class

Application in a separate file

Object Reference

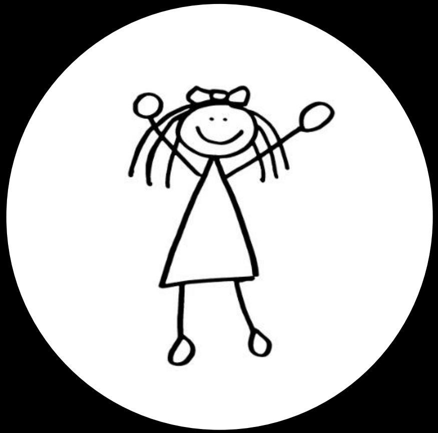
- Declaring a variable creates a **reference** in the stack
  - **new** keyword allocates memory on the heap

```
Person Eli = new Person();
```



Classes vs. Objects

- Classes provide **structure** for describing and creating objects
- An **object** is a **single instance of a class**



Person
(Class)

Eli
(Object)

Built-in API Classes

- Java provides ready-to-use classes
  - Bundled into **packages** like `java.lang`, `java.io`, `java.util`, etc.
- Using static class members:

```
int num = Integer.parseInt("3,14");
double cosine = Math.cos(Math.PI);
```

`Class.StaticMember`

- Using non-static classes

```
Random rnd = new Random();
int randomNumber = rnd.nextInt();
```

`new Class(…)`

`Object.Member`

Collections API

SOFTWARE UNIVERSITY FOUNDATION

- **Collections API** provides functionality for storing, retrieving and manipulating **sequences of elements**

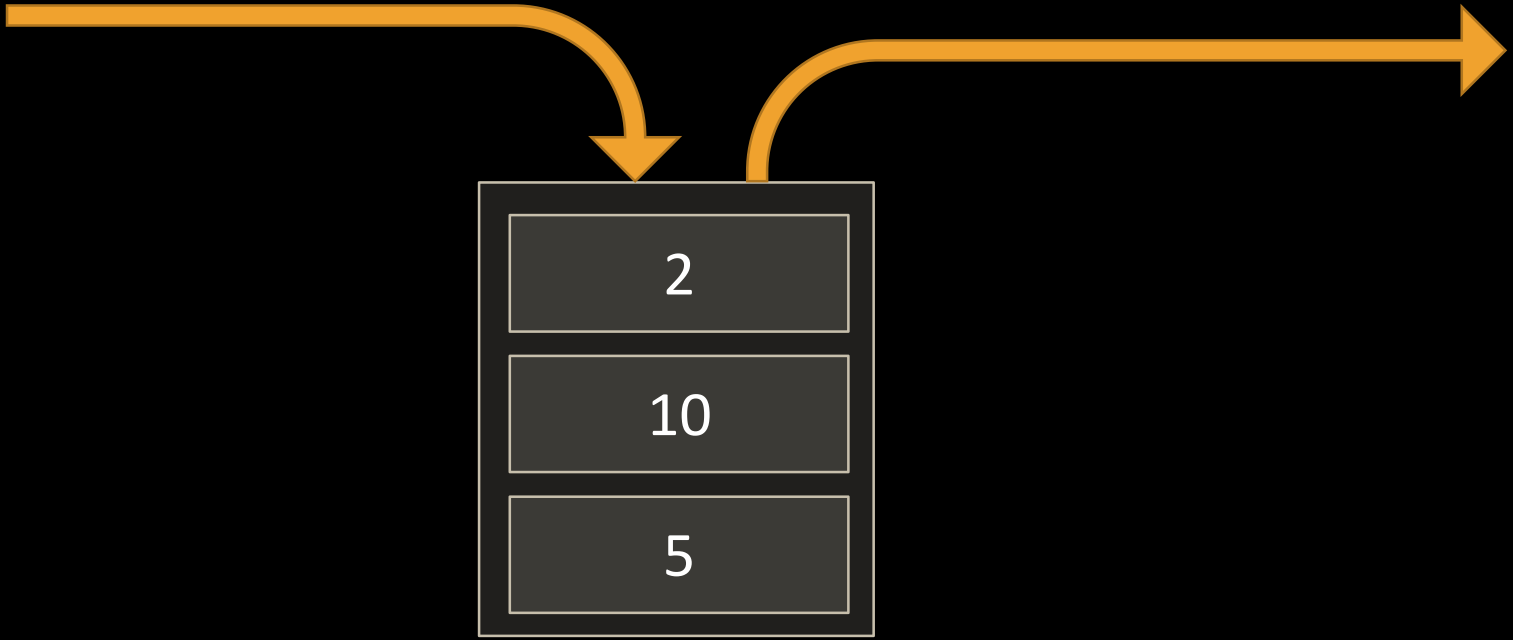
```java
ArrayList<String> names = new ArrayList<>();
names.add("Pesho");
Collections.addAll(names, "Gosho", "Mariika", "Ivancho");

Collections.sort(names);
Collections.reverse(names);
names.remove("Pesho");

names.clear();
```

[Gosho, Ivancho, Mariika, Pesho]

[Pesho, Mariika, Ivancho, Gosho]

[Mariika, Ivancho, Gosho]

[]

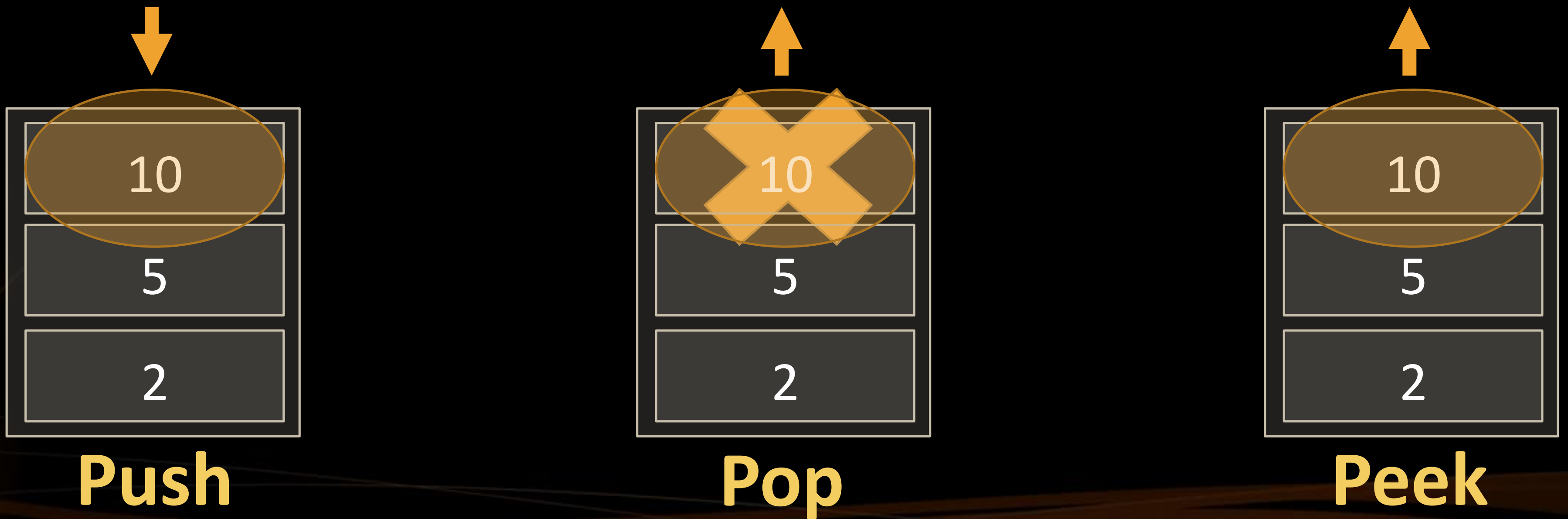


Stack

Last In First Out

Stack – Abstract Data Type

- **Stacks** provide the **following functionality:**
  - Pushing an element at the **top** of the stack
  - Popping element from the **top** fo 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);
```

ArrayDeque<E> – Java Stack Implementation (2)

- Removing elements

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

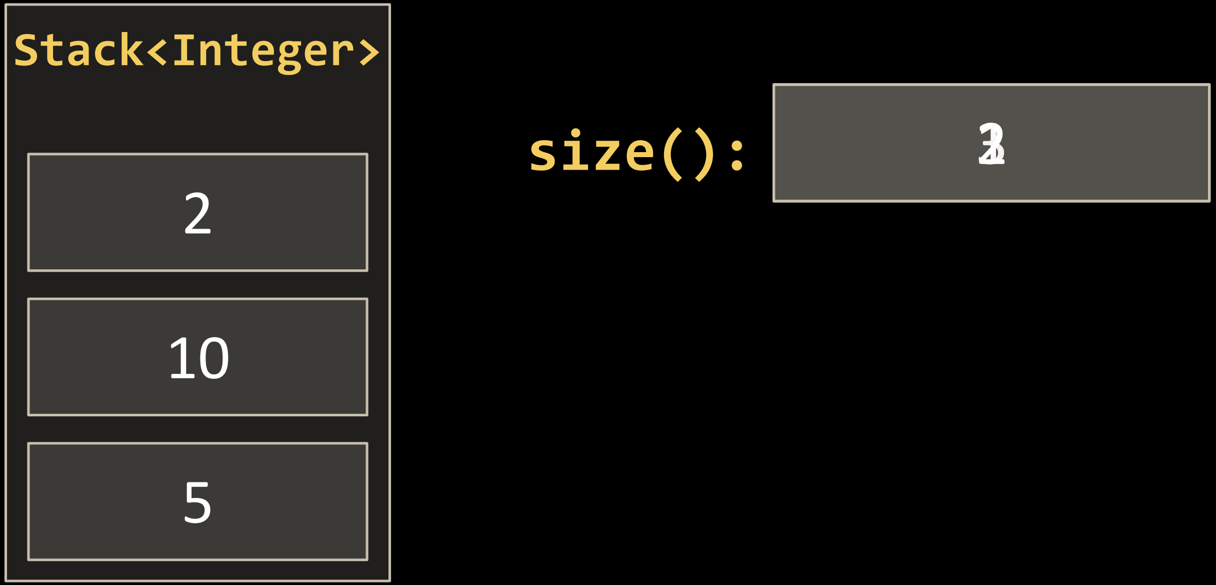
- Getting the value of the topmost element

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

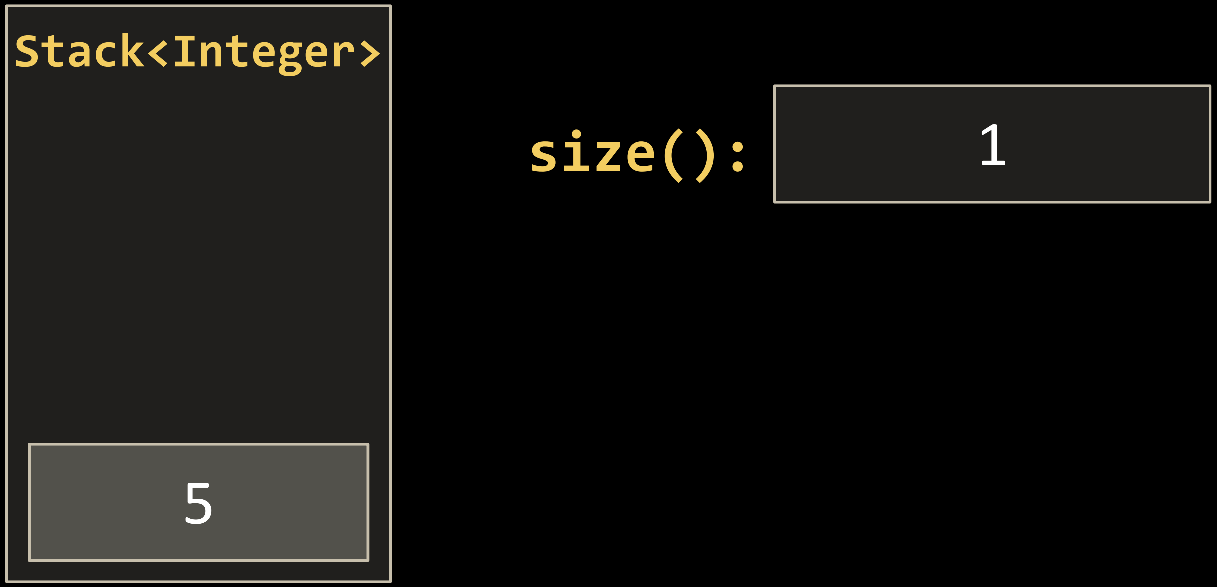
push() – Adds an element on top of the Stack

15
Stack<Integer>
size():
0

pop() – Returns the last element from the stack and removes it



peek() – Returns the last element from the stack, but **does not** remove it



Stack – Utility Methods

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

Integer size = stack.size();
boolean isEmpty = stack.isEmpty();
boolean exists = stack.contains(2);
Integer[] arr = stack.toArray();
```

Retains the order of elements

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
```

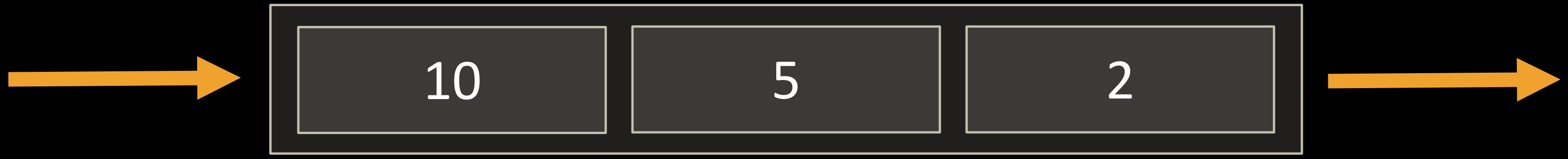
```
(2 + 3)
(3 + 1)
(2 - (2 + 3) * 4 / (3 + 1))
```

Check your solution here: https://judge.softuni.bg/Contests/781

Problem: Matching Brackets

```java
// TODO: Initialize the stack

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/781

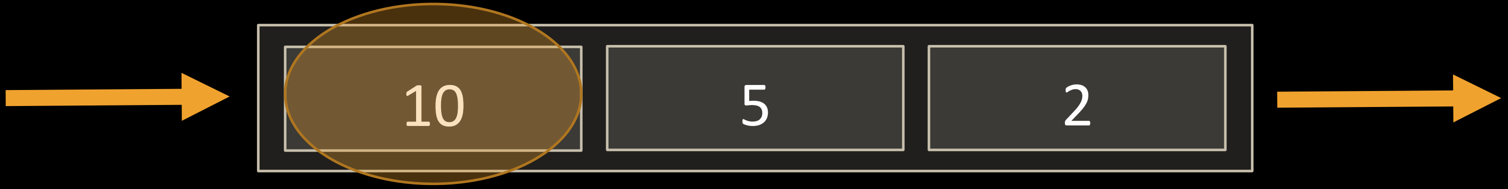


Queue

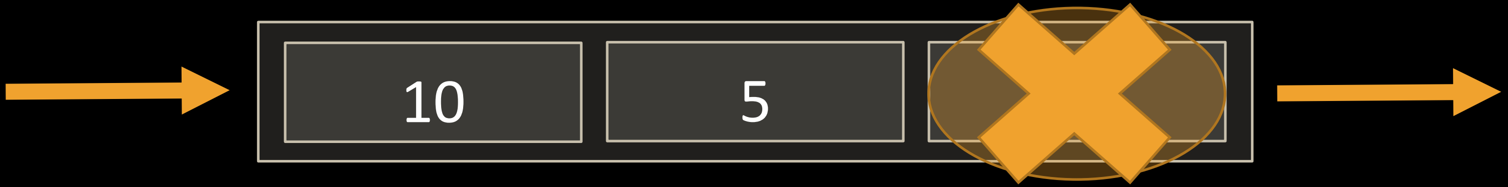
First In First Out

Queue – Abstract Data Type

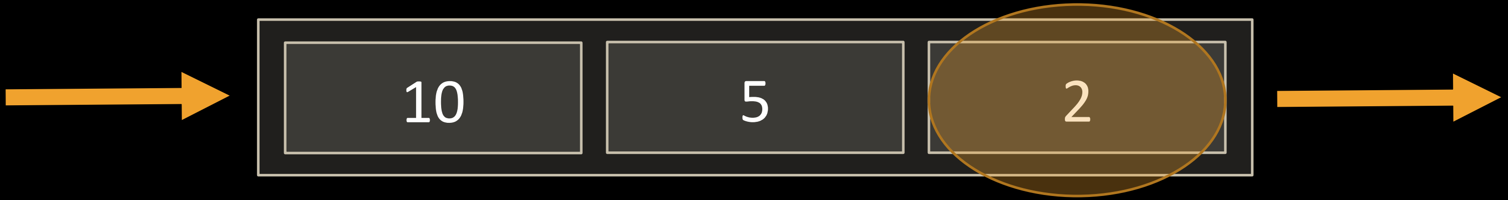
- **Queues** provide the **following functionality**:
  - Adding an element at the end of the queue



  - Removing the first element from the queue



  - Getting the first element of the queue without removing it



ArrayDeque<E> – Java Queue Implementation

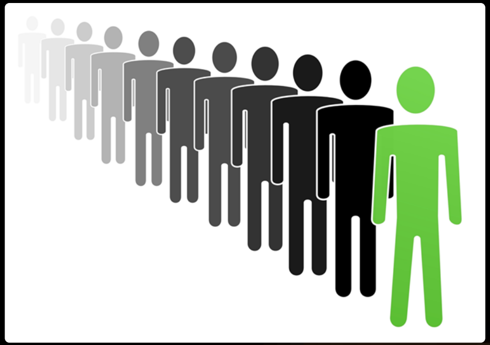
- Creating a Queue

```
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)

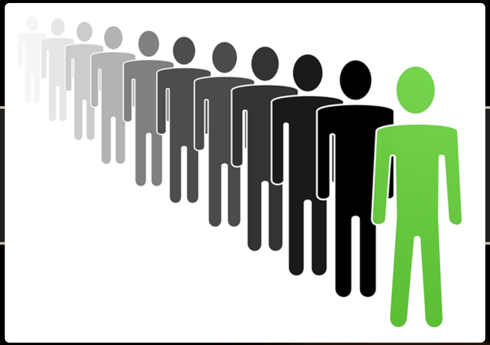
- Removing elements

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

  - `remove()` – throws exception if queue is empty
  - `poll()` – returns null if queue is empty

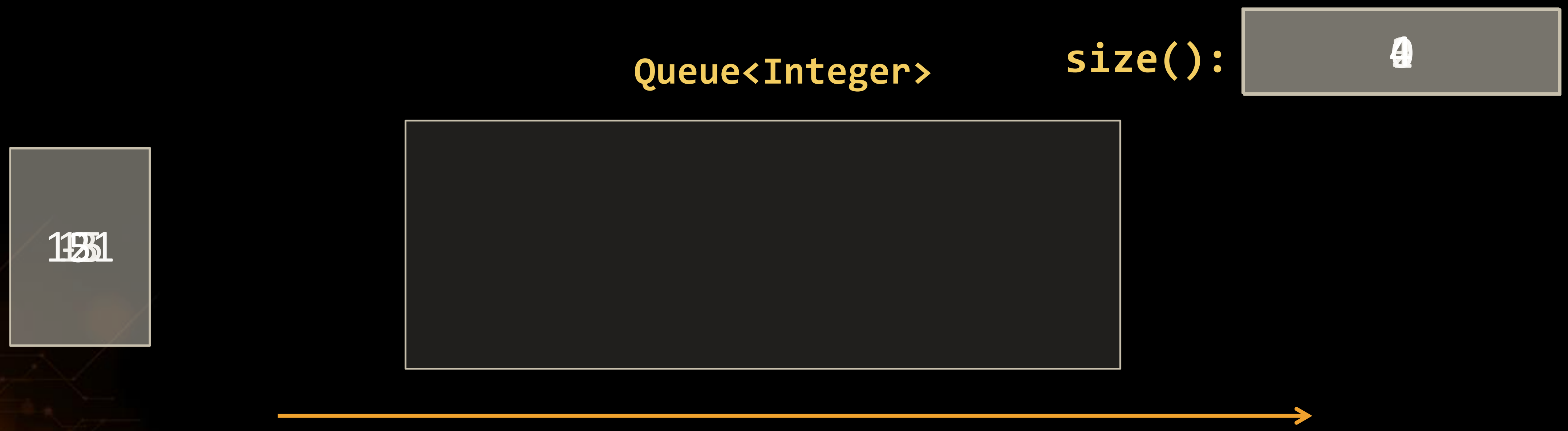
- Check first element

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



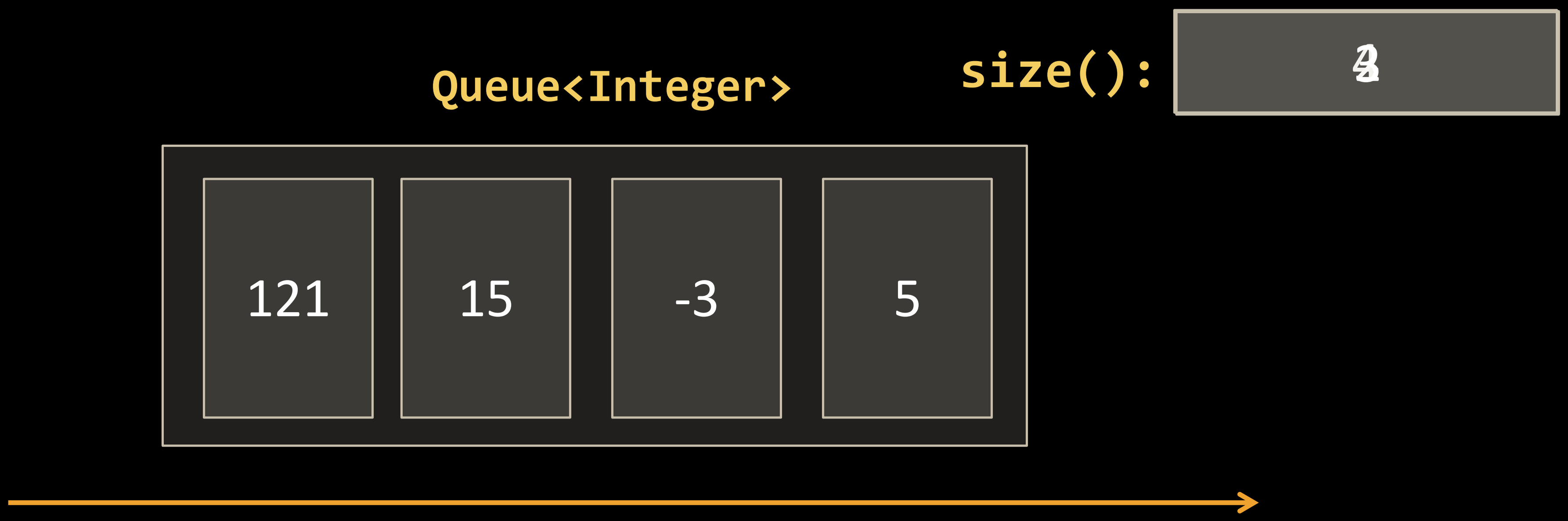
add() / offer()

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

```
Mimi Pepi Toshko
2
```

$\rightarrow$

```
Removed Pepi
Removed Mimi
Last is Toshko
```

Check your solution here: https://judge.softuni.bg/Contests/781

Solution: Hot Potato (2)

```
// TODO: Initialize the queue and add children

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/781

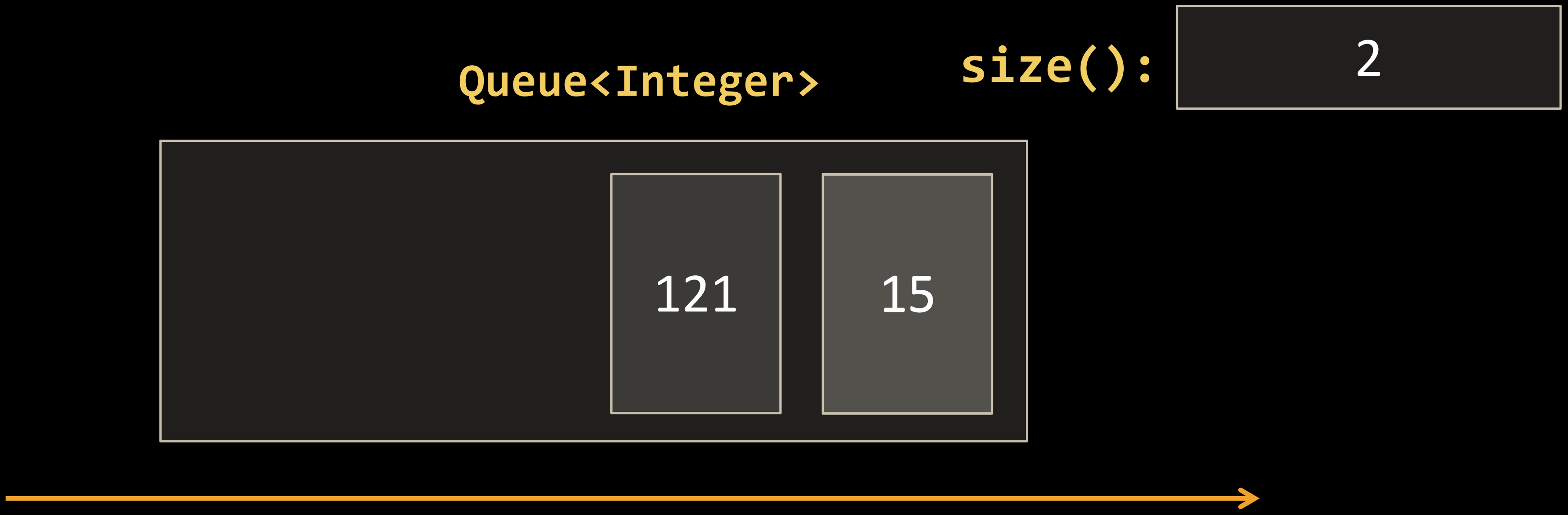
Queue – Utility Methods

- `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

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

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

```
Mimi Pepi Toshko
2
```

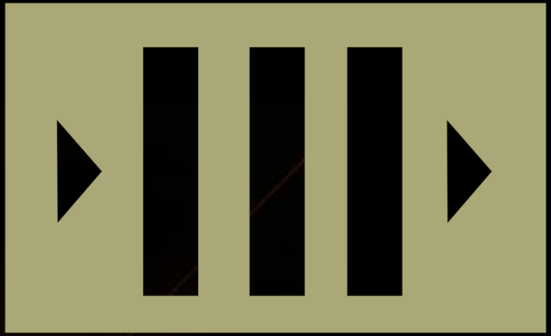
```
Removed Pepi
Prime Mimi
Prime Toshko
Removed Mimi
Last is Toshko
```

Check your solution here: https://judge.softuni.bg/Contests/781

Solution: Math Potato

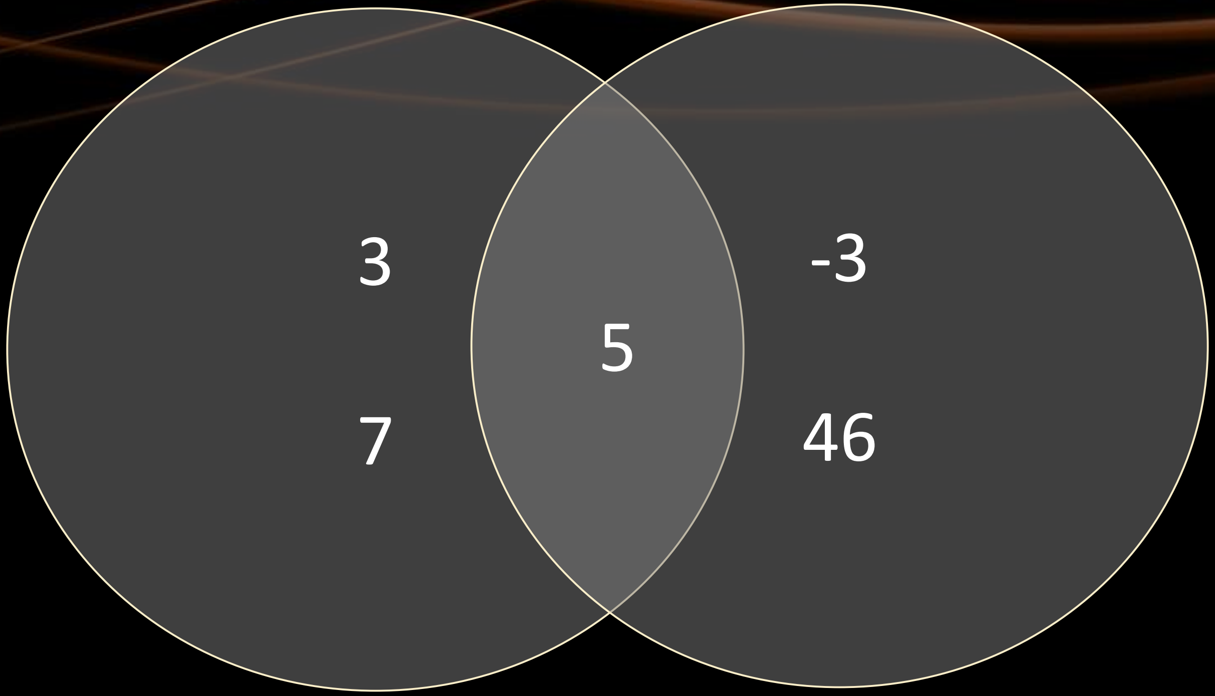
```
int cycle = 1;
while (queue.size() > 1) {
  for (int i = 1; i < n; i++)
    queue.offer(queue.poll());

  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/781



Practice: Working with Stacks and Queues

Live Exercises in Class (Lab)



Sets

`HashSet<E>`, `TreeSet<E>` and `LinkedHashSet<E>`

Sets in Java

- A **Set** keeps unique elements
  - Provides methods for adding/removing/searching elements
  - Offers very fast performance
- Initialization

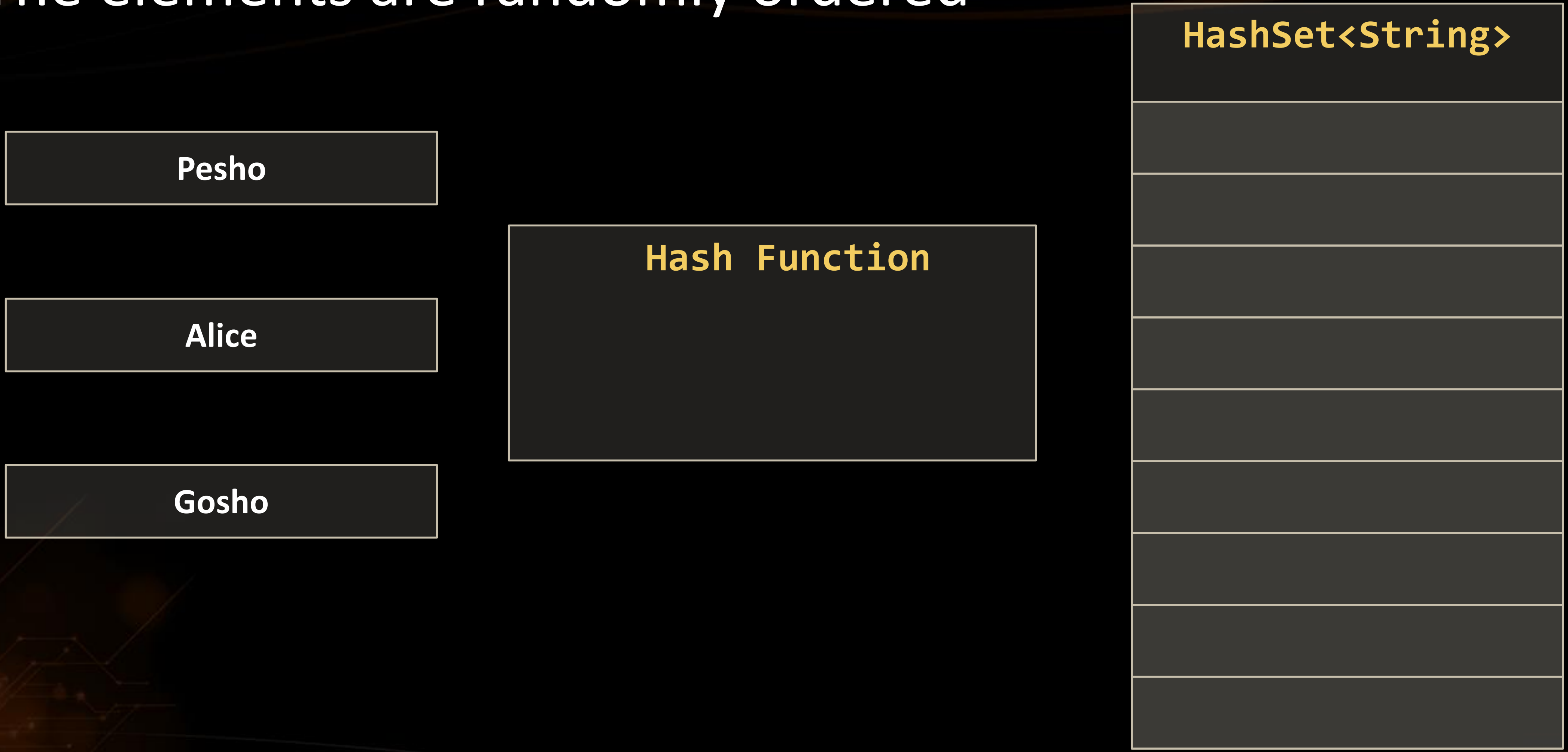
```java
HashSet<String> hash = new HashSet<>();
```

- `.size()` & `.isEmpty()`

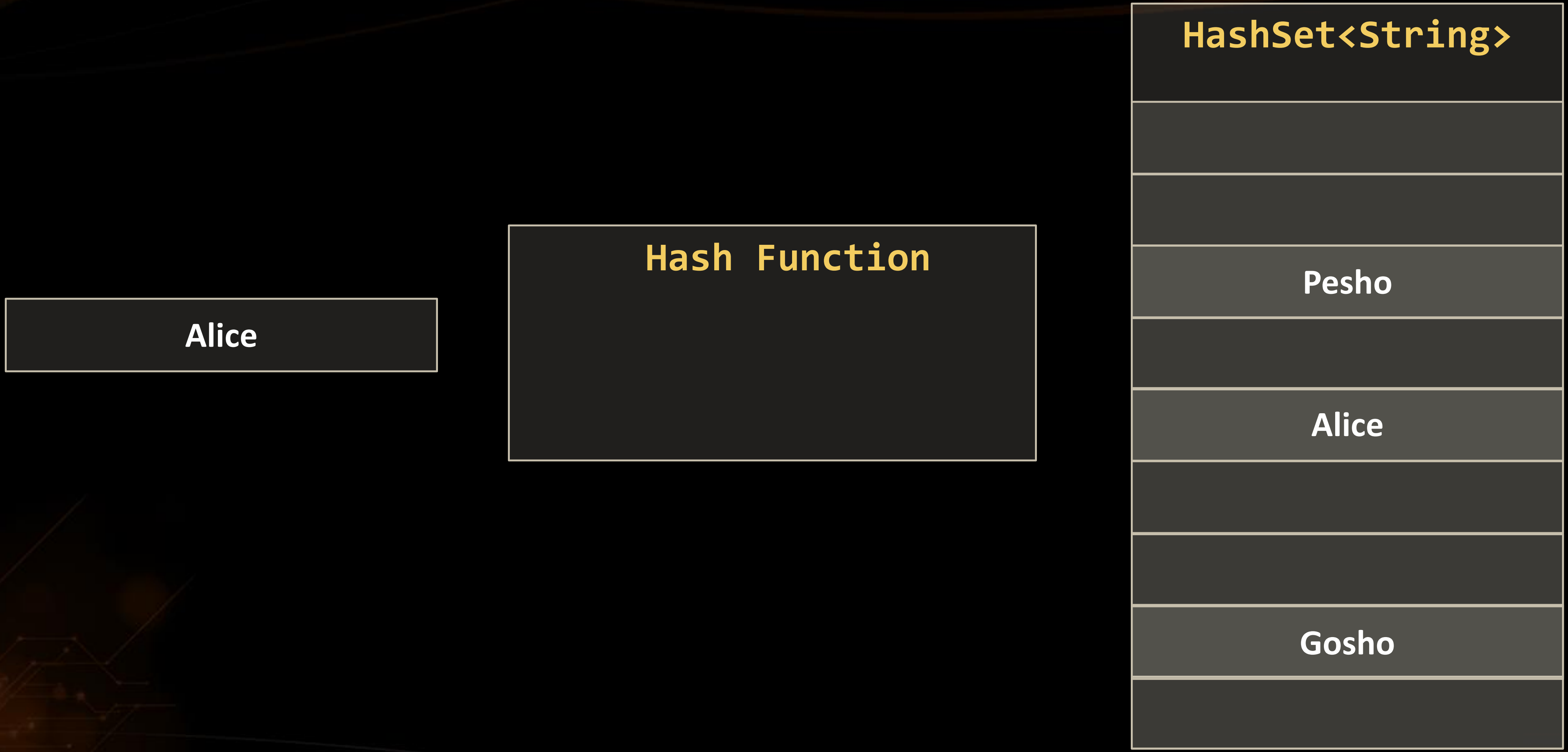
```java
System.out.println(hash.size()); // 0
System.out.println(hash.isEmpty()); // True
```

HashSet<E> – add()

- The elements are randomly ordered

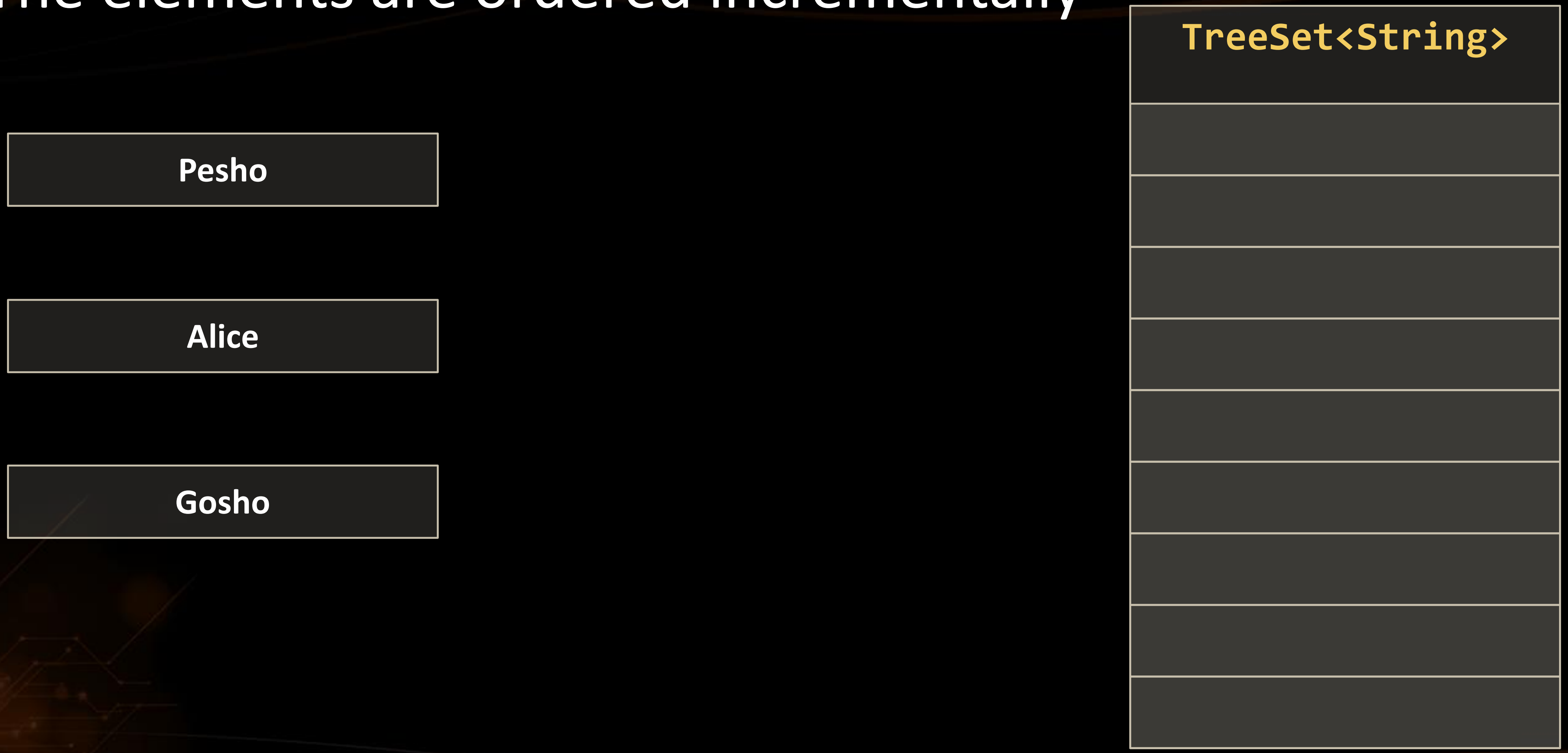


HashSet<E> – remove()



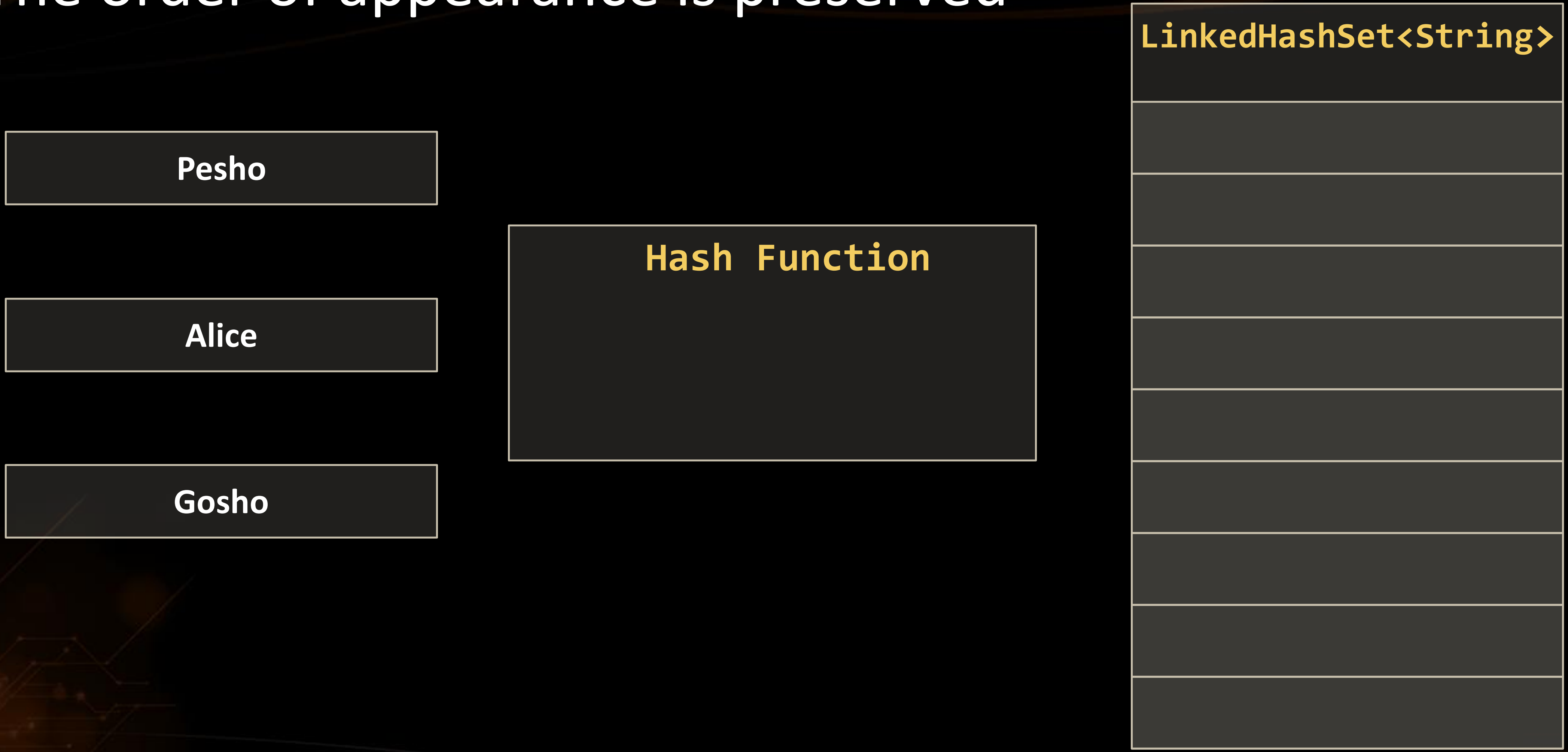
TreeSet<E> – add()

- The elements are ordered incrementally



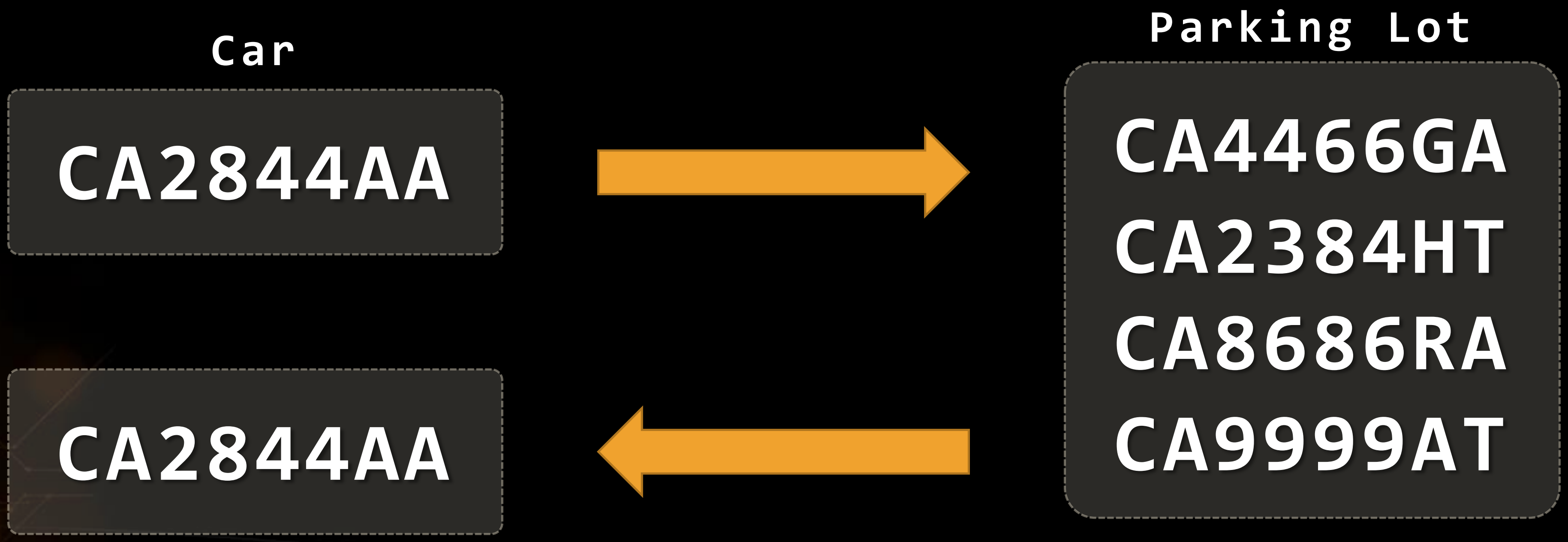
LinkedHashSet<E> – add()

- The order of appearance is preserved



Problem: Parking Lot

- Write a program that:
  - Records car number for every car that enters a parking lot
  - Removes car number when the car goes out



Check your solution here: https://judge.softuni.bg/Contests/781

Solution: Parking Lot

```
HashSet<String> parkingLot = new HashSet<String>();
while(true)
  String input = sc.nextLine();
  if (input.equals("END"))
    break;
  else
    String[] reminder = input.split(", ");
    if (reminder[0].equals("IN"))
      parkingLot.add(reminder[1]);
    else
      parkingLot.remove(reminder[1]);
```

Check your solution here: https://judge.softuni.bg/Contests/781

Problem: SoftUni party

- Two types of guests:
  - Regular
  - VIP – their tickets starts with a **digit**
- First you will receive the **invited guests**
- Then you will receive the **guests who came**
- Find how many guests didn't come to the party
- Print all guests that **didn't come** (VIP first)

Reservation List

```
7IK9Yo0h
9NoBUajQ
Ce8vwPmE
SVQXQCbc
```

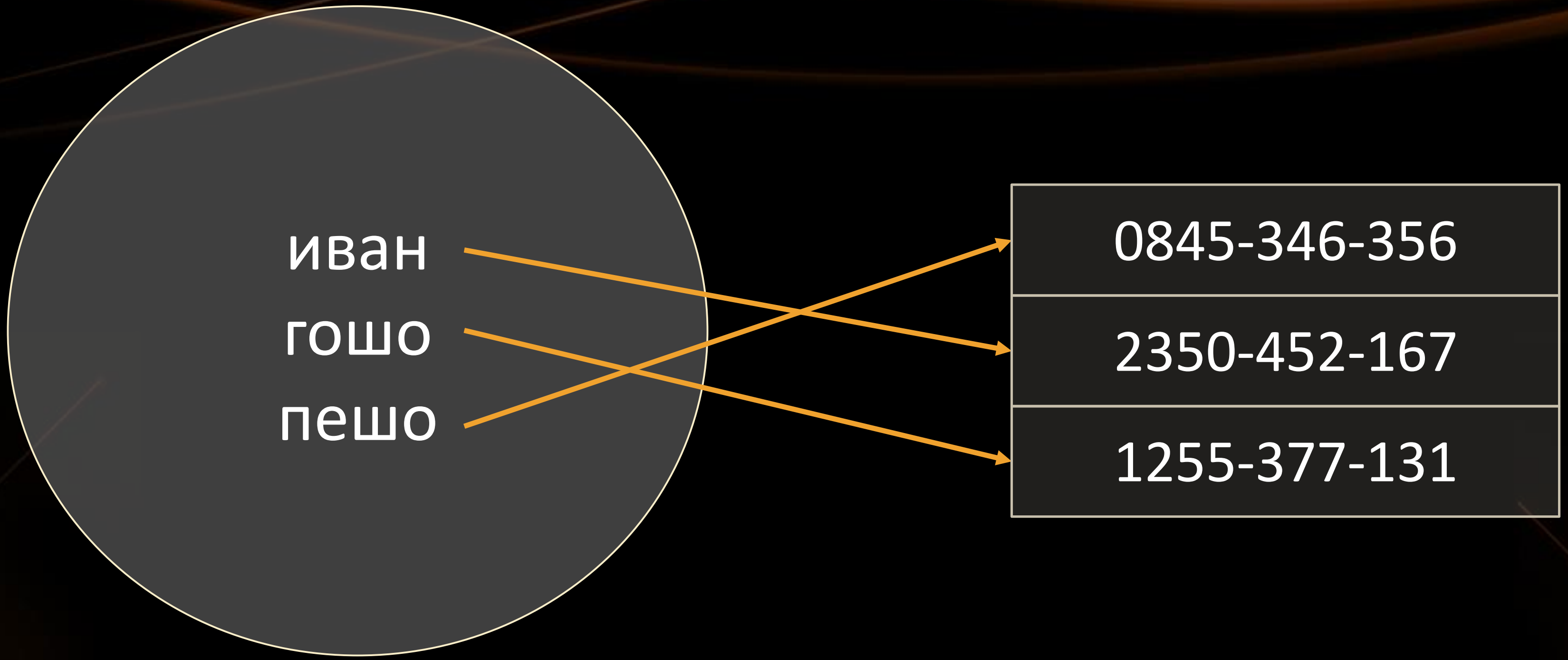
Check your solution here: https://judge.softuni.bg/Contests/781

Solution: SoftUni party

```
HashSet<String> vip = new HashSet<String>();
TreeSet<String> regular = new TreeSet<String>();
while (true)
  String input = sc.nextLine();
  if (input.equals("PARTY")) break;
  else
    String sign = Character.toString(input.charAt(0));
    if (numbers.contains(sign))
      vip.add(input);
    else
      regular.add(input);
//TODO: Remove from guest, that came to party
regular.addAll(vip);
//TODO: Print results
```

Returns `true` or `false`

Check your solution here: https://judge.softuni.bg/Contests/781



Associative Arrays

`HashMap<Key, Value>`

Associative Arrays (Maps)

- **Associative arrays** are arrays indexed by keys
  - Not by the indexes 0, 1, 2, …
- Hold a set of **pairs `<key, value>`**

- Traditional array

key	0	1	2	3	4
value	8	-3	12	408	33

- Associative array

key	value
John Smith	+1-555-8976
Lisa Smith	+1-555-1234
Sam Doe	+1-555-5030

Maps Methods

- Initialization

```
HashSet<String, Integer> hash = new HashSet<String>();
```

Type of keys

Type of values

- .size()
- .isEmpty()

```
HashSet<String> hash = new HashSet<>();
System.out.println(hash.size()); // 0
System.out.println(hash.isEmpty()); // True
```

HashMap<K, V> – put()

Pesho	[illegible]

Hash Function

HashMap<String, String>

Key	Value

HashMap<K, V> – remove()

Pesho

Hash Function

HashMap<String, String>	
Gosho	0881-456-987
Pesho	0881-123-987
Alice	+359-899-55-592
Key	Value

Looping Through Maps - Example

```
HashMap<String, Integer> vehicles = new HashMap<>();
vehicles.put("BMW", 5);
vehicles.put("Mercedes", 3);
vehicles.put("Audi", 4);
vehicles.put("BMW", 10);
for(String key: vehicles.keySet())
  System.out.println(key + " - " + vehicles.get(key));
```

Override first value

Return set of all keys

Return value for key

```
Audi - 4
Mercedes - 3
BMW - 10
```

TreeMap<K, V> – put()

Pesho Alice	[illegible]

Tree Map<String, String>	
Key	Value

Problem: Academy Graduation

- Write a program that:
  - Reads a list of **students** and their **score** for some courses
  - Prints a **sorted** list with **average** score for each student

Student	Java Advanced	Java OOP
Gosho	3.75	5
Mara	4.25	6
Pesho	6	4.5

Student	Average
Gosho	4,375
Mara	5,125
Pesho	7,25

Check your solution here: https://judge.softuni.bg/Contests/781

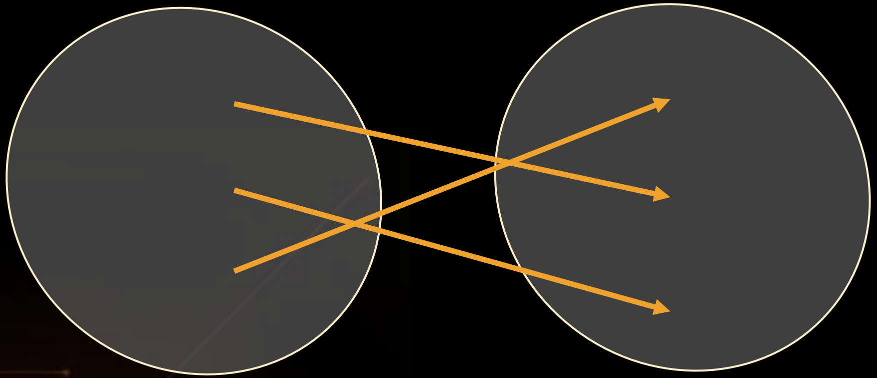
Solution: Count Same Values in Array

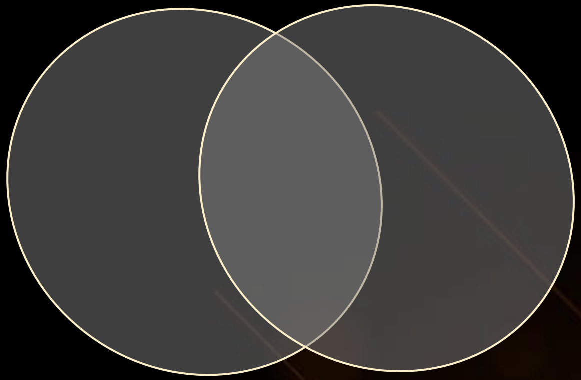
```java
TreeMap <String,Double[]> graduationList = new TreeMap<>();
for (int i = 0; i < numberOfStudents; i++) {
  String name = scanner.nextLine();
  String[] scoresStrings = scanner.nextLine().split(", ");
  Double[] scores = new Double[scoresStrings.length];

  for (int j = 0; j < scoresStrings.length; j++) {
    scores[j] = Double.parseDouble(scoresStrings[j]);
  }
  graduationList.put(name, scores);
}
//TODO: print results
```

Check your solution here: https://judge.softuni.bg/Contests/781

Maps - Utility Methods

- `size()` – the number of key-value pairs
- `keySet()` – a set of unique keys
- `values()` – a collection of all values
- Basic operations – `put()`, `remove()`, `clear()`
- Boolean methods:
  - `containsKey()` – checks if a key is present in the dictionary
  - `containsValue()` – checks if a value is present in the dictionary





Practice: Working with Sets and Maps

Live exercises in class (Lab)

Summary

- **Classes** provide **structure** for describing and creating objects
- **Object** is a **single instance of a class**
- **Stack<E>** – **LIFO** data structure
  - The last element that is put in the stack is the first to come out
- **Queue<E>** – **FIFO** data structure
  - The first element that is put in the queue is the first to come out
- **Sets** hold unique elements and are very fast
- **Maps** are associative arrays where a **value** is accessed by its **key**

Objects, Classes, Collections

Questions?

https://softuni.bg/opencourses/algorithms

License

Free Trainings @ Software University

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