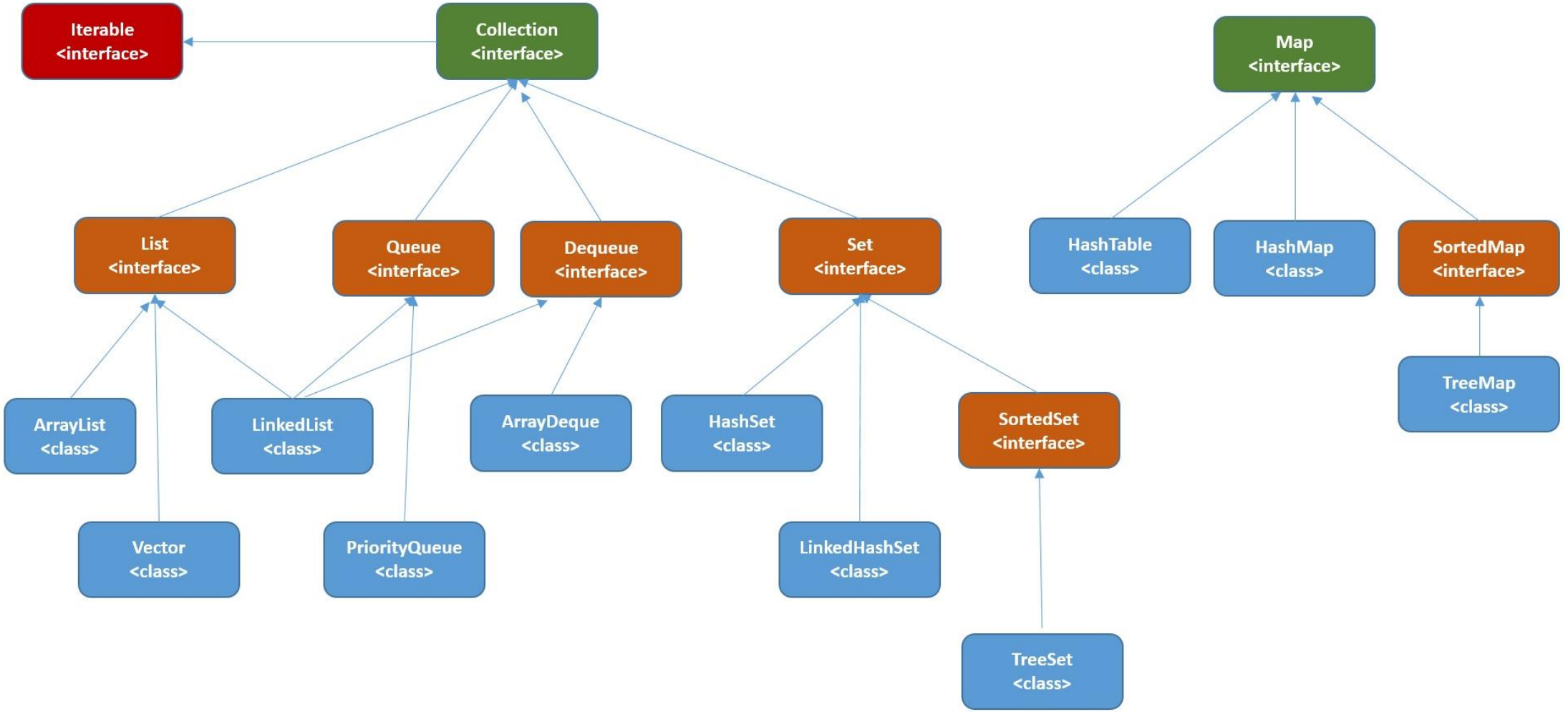
12. Lekcja

Kolekcje

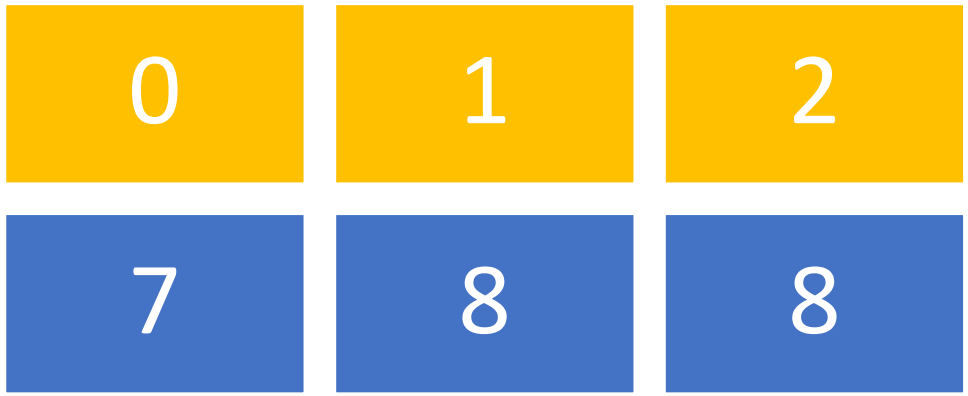
Co to jest kolekcja ?

- Kolekcją jest pojedynczy obiekt który pozwala na przechowywanie w nim wielu elementów.
- Kolekcje są „kontenerem” do przechowywania innych obiektów w Javie
- W odróżnieniu od tablicy kolekcje są bardziej zaawansowane oraz elastyczne – podczas gdy tablice oferują przechowywanie określonej ilości danych, kolekcje przechowują dane dynamiczne.

Hierarchia kolekcji



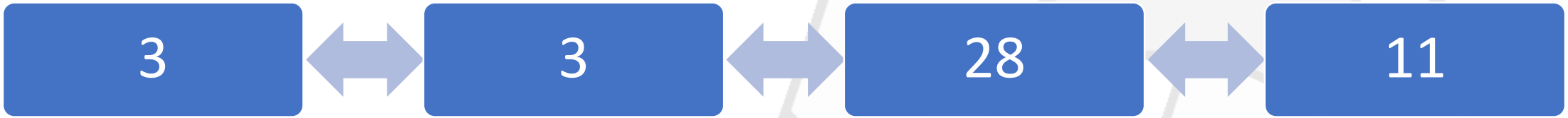
Tablice dynamiczne (Array)



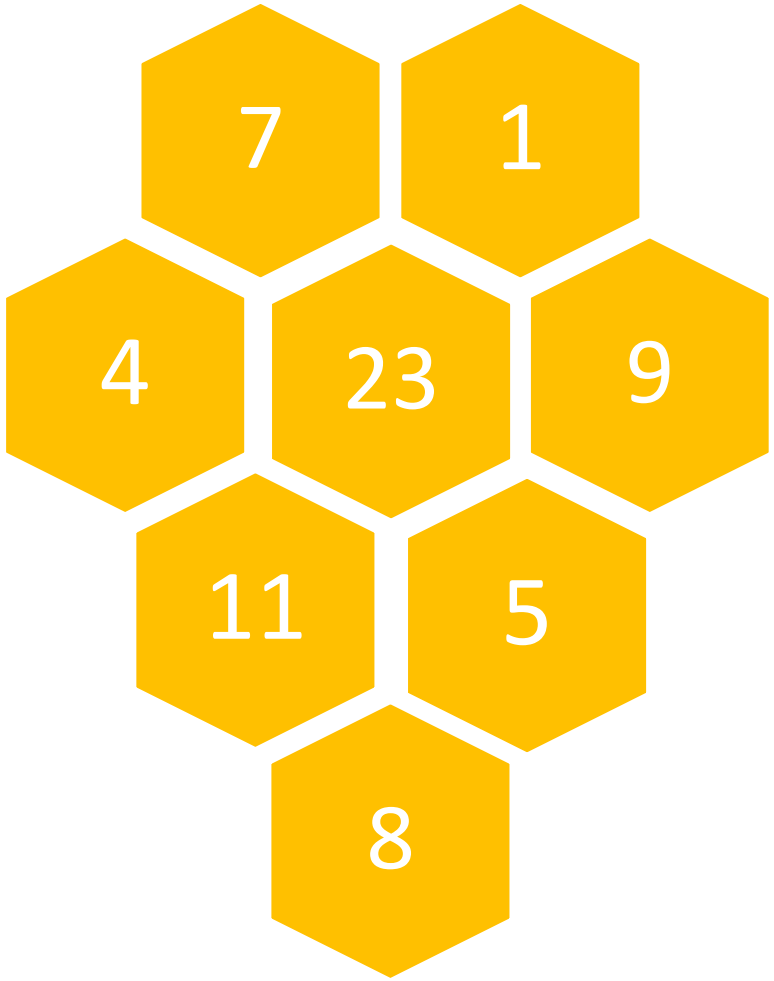
- To reprezentacja standardowej tablicy tylko bez deklarowania jej początkowej długości.
- Do elementów dostajemy się poprzez wskazanie indeksu.
- Może posiadać duplikujące się elementy.
- W momencie gdy pamięć pierwotnie zaalokowana kończy się, tablica dynamiczna automatycznie zwiększa swoją długość przepisując wszystkie elementy tablicy do 2 razy większej tablicy.

Lista (List)

- Do elementów dostajemy się poprzez przesuwanie się po każdym elemencie listy aż dojdziemy do odpowiedniego.
- Może posiadać duplikujące się elementy.
- Pamięć alokowana jest w momencie dodawania nowego elementu i zawsze tylko tyle ile elementów dodajemy. Elementy dodawane są zawsze na koniec listy.



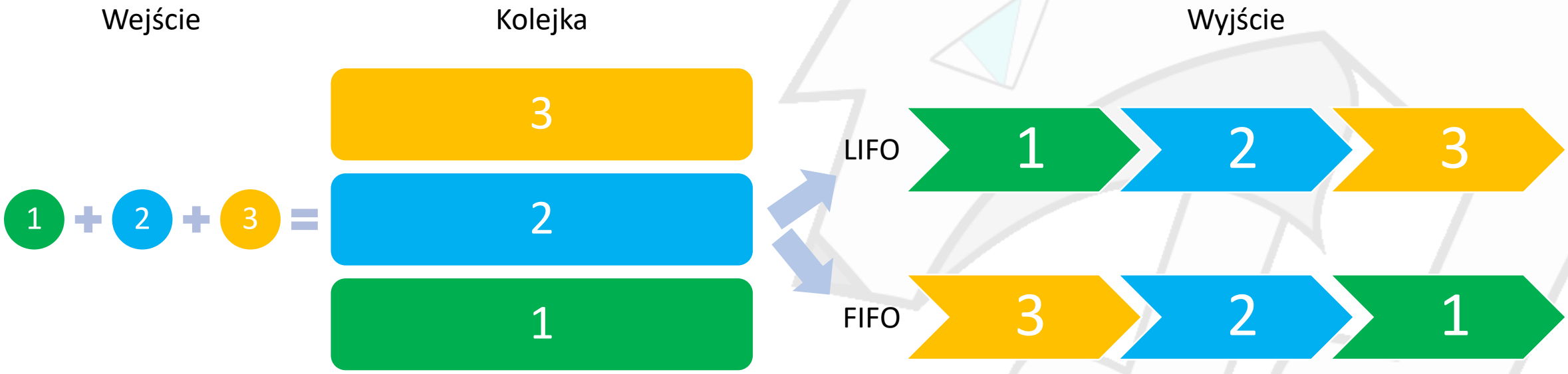
Zbiór (Set)



- Zbiór nie jest kolekcją mającą zachowaną kolejność. Do elementów dostajemy się poprzez przesuwanie się po każdym elemencie aż znajdziemy odpowiedni.
- Nie może posiadać duplikujących się elementów.

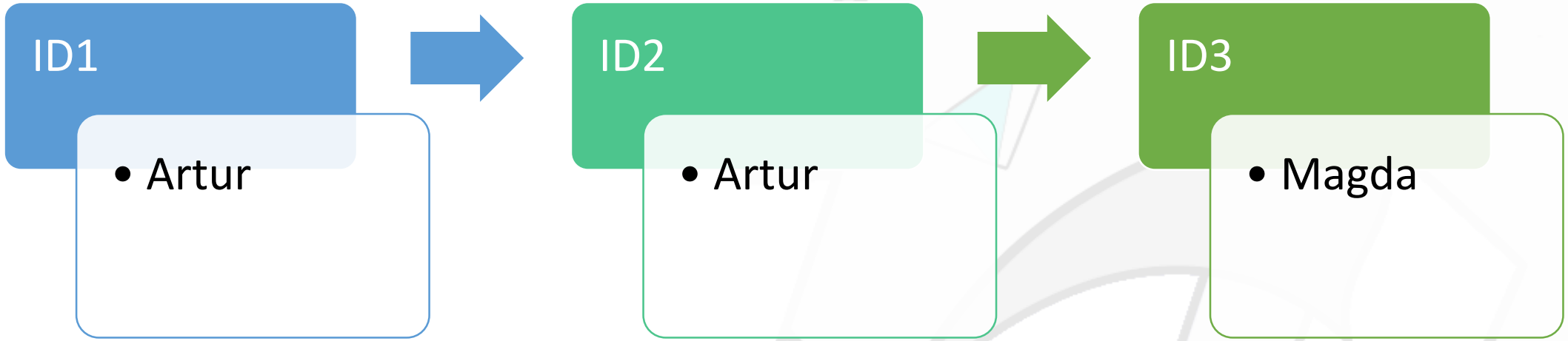
Kolejka i sterta (Queue and Stack)

- Pozwala na przechowywanie elementów w kolejności do przetworzenia
- Elementy możemy odczytywać tylko z początku lub końca kolejki po kolei zgodnie ze strategiami LIFO (last in, first out) lub FIFO (first in, first out).
- Może posiadać duplikujące się elementy.



Mapa (Map)

- To obiekty które przechowują relację klucze -> wartości.
- Nie może posiadać dwóch takich samych kluczy – może natomiast posiadać dwie takie same wartości (pod warunkiem że mają różne klucze)



Deklaracja i inicjalizacja kolekcji

nazwa_klasy_kolekcji <klasa_obiektów_przechowywanych>
nazwa_zmiennej = new nazwa_klasy_kolekcji<>()

```
ArrayList<Object> objects = new ArrayList<>();
LinkedList<Exception> exceptions = new LinkedList<>();
HashSet<Date> birthDays = new HashSet<>();
PriorityQueue<Files> files = new PriorityQueue<>();
```

Deklaracja i inicjalizacja mapy

nazwa_klasy_mapy < klasa_klucza, klasa_obiektów_przechowywanych>
nazwa_zmiennej = new nazwa_klasy_mapy <>()

```
HashMap<String, Integer> grades = new HashMap<>();
```

Operacje na kolekcjach

- **dodawanie elementów** - dodawanie elementów w liście jest łatwe i szybkie w porównaniu do tablicy dynamicznej gdyż nie istnieje ryzyko powiększenia tablicy i kopiowania zawartości do nowej tablicy.
- **odczyt elementu** – najszybciej działa w przypadku tablic dynamicznych ze względu na posiadany indeks, następnie w kolejkach i stertach. Pod względem optymalizacji najgorzej wypada w listach.
- **usuwanie elementu** – najkorzystniej wypada tutaj lista ze względu na łatwość w przepięciu elementów. Później tablica dynamiczna, chyba że usuwamy element z końca. W przypadku usuwania ze środka wszystkie elementy musza zostać przepisane do nowej tablicy.

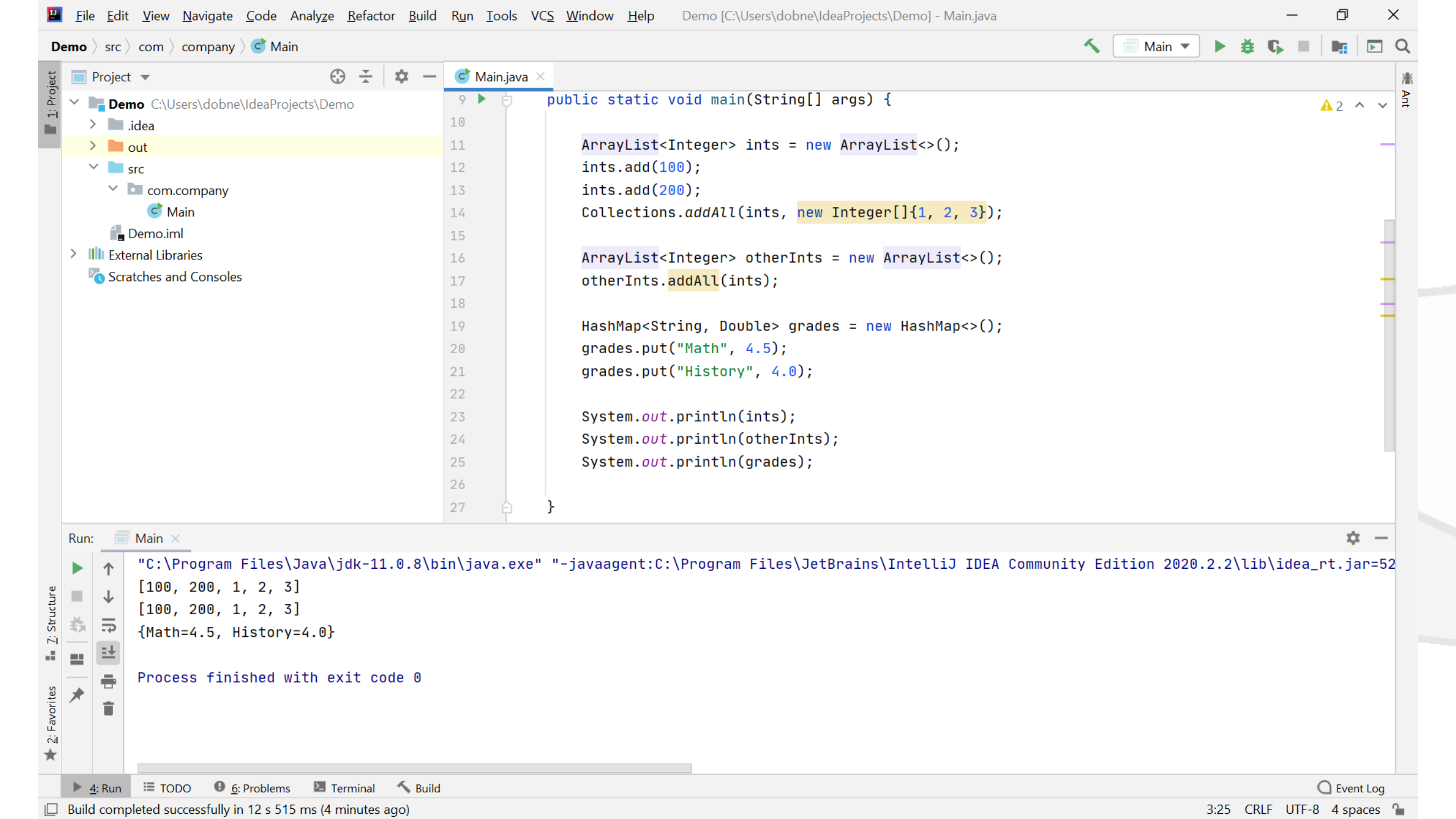
Dodawanie elementów

- W celu dodania elementu do kolekcji służy metoda „add()” która dodaje jeden element do kolekcji lub „addAll()”, która służy do dodawania większej liczby elementów.
- W celu dodawania do kolekcji zwykłych tablic możemy wykorzystać statyczną metodę „addAll()” w klasie Collections.
- Do dodawania elementów do map wykorzystujemy metodę „put()”

```
ArrayList<Integer> ints = new ArrayList<>();
ints.add(100);
ints.add(200);
Collections.addAll(ints, new Integer[]{1, 2, 3});

ArrayList<Integer> otherInts = new ArrayList<>();
otherInts.addAll(ints);

HashMap<String, Double> grades = new HashMap<>();
grades.put("Math", 4.5);
grades.put("History", 4.0);
```


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Demo [C:\Users\dobne\IdeaProjects\Demo] - Main.java
Demo src com company Main
Main
Project
Demo C:\Users\dobne\IdeaProjects\Demo
.idea
out
src
com.company
Main
Demo.iml
External Libraries
Scratches and Consoles
Main.java
public static void main(String[] args) {
ArrayList<Integer> ints = new ArrayList<>();
ints.add(100);
ints.add(200);
Collections.addAll(ints, new Integer[]{1, 2, 3});
ArrayList<Integer> otherInts = new ArrayList<>();
otherInts.addAll(ints);
HashMap<String, Double> grades = new HashMap<>();
grades.put("Math", 4.5);
grades.put("History", 4.0);
System.out.println(ints);
System.out.println(otherInts);
System.out.println(grades);
}
Run: Main
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.2\lib\idea_rt.jar=52
[100, 200, 1, 2, 3]
[100, 200, 1, 2, 3]
{Math=4.5, History=4.0}
Process finished with exit code 0
4: Run
TODO
6: Problems
Terminal
Build
Event Log
Build completed successfully in 12 s 515 ms (4 minutes ago)
3:25 CRLF UTF-8 4 spaces

Odczyt elementu

```
ArrayList<Date> dates = new ArrayList<>();
System.out.println(dates.get(0));

LinkedList<Integer> years = new LinkedList<>();
years.getFirst();
years.get(1);
years.getLast();

HashMap<String, Double> grades = new HashMap<>();
grades.get("History");

HashSet<Boolean> booleans = new HashSet<>();
for (Boolean b : booleans) {
    System.out.print(b + " ");
}
```

- Do odczytu elementu o podanym indeksie służy metoda „get()”.
- Listy dwukierunkowe posiadają dodatkowo możliwość pobrania pierwszego i ostatniego elementu z listy przez metody „getFirst()” i „getLast()”
- Wartości z map pobieramy przez podanie odpowiedniego klucza w metodzie „get()
- Ze zbiorów nie jesteśmy w stanie pobierać wartości inaczej niż iterując się jeden po drugim.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Demo [C:\Users\dobne\IdeaProjects\Demo] - Main.java

Demo > src > com > company > Main > main

Main

Project

Demo C:\Users\dobne\IdeaProjects\Demo
.idea
out
src
com.company
Main
Demo.iml
External Libraries
Scratches and Consoles

Main.java

```
ArrayList<Date> dates = new ArrayList<>();
dates.add(new Date());
System.out.print("Element with index 0 is ");
System.out.println(dates.get(0));

LinkedList<Integer> years = new LinkedList<>();
Collections.addAll(years, new Integer[]{2012, 2021, 2021, 2030});
System.out.println("First element is " + years.getFirst());
System.out.println("Element with index 1 is " + years.get(1));
System.out.println("Last element is " + years.getLast());

HashMap<String, Double> grades = new HashMap<>();
grades.put("Math", 4.5);
grades.put("History", 4.0);
System.out.println("Value for key History is " + grades.get("History"));

HashSet<Boolean> booleans = new HashSet<>();
booleans.add(true);
booleans.add(true);
booleans.add(false);
System.out.print("Set contains elements ");
for (Boolean b : booleans) {
    System.out.print(b + " ");
```

Run: Main

```
Element with index 0 is Sun Aug 08 00:06:19 CEST 2021
First element is 2012
Element with index 1 is 2021
Last element is 2030
Value for key History is 4.0
Set contains elements false true
```

4: Run TODO 6: Problems Terminal Build Event Log

Build completed successfully in 8 s 818 ms (3 minutes ago) 17:41 CRLF UTF-8 4 spaces

Usuwanie elementów

```
ArrayList<File> files = new ArrayList<>();
files.clear();
files.remove( index: 0);
files.remove(new File( pathname: "file.txt"));

HashMap<String, Double> grades = new HashMap<>();
grades.clear();
grades.remove( key: "History");
```

- Do wyczyszczenia całej kolekcji służy metoda „clear()”;
- Do usuwania elementów służy metoda „remove()”. Jako parametr przyjmuje indeks usuwanego elementu lub sam usuwany obiekt.
- Usuwanie elementów z mapy wykonuje się przez podanie w metodzie „remove()” odpowiedniego klucza.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Demo [C:\Users\dobne\IdeaProjects\Demo] - Main.java

Demo > src > com > company > Main > main

Main

Project

Demo C:\Users\dobne\IdeaProjects\Demo
- .idea
- out
- src
  - com.company
    - Main
  - Demo.iml
- External Libraries
- Scratches and Consoles

Main.java

```
8
9       ArrayList<Integer> ints = new ArrayList<>();
10      Collections.addAll(ints, new Integer[]{1, 2, 3});
11      System.out.println(ints);
12      ints.remove( index: 1);
13      System.out.println(ints);
14
15      HashSet<String> names = new HashSet<>();
16      names.add("Artur");        names.add("Maciek");        names.add("Magda");        names.add("Artur");
17      System.out.println(names);
18      names.remove( o: "Artur");
19      System.out.println(names);
20      names.clear();
21      System.out.println(names);
22
23      HashMap<String, Double> grades = new HashMap<>();
24      grades.put("Math", 4.5);        grades.put("History", 4.0);
25      System.out.println(grades);
26      grades.remove( key: "History");
27      System.out.println(grades);
28
```

3 1

Run: Main

```
[1, 2, 3]
[1, 3]
[Artur, Maciek, Magda]
[Maciek, Magda]
[]
{Math=4.5, History=4.0}
{Math=4.5}
```

4: Run TODO 6: Problems Terminal Build Event Log

Build completed successfully in 8 s 762 ms (moments ago) 8:1 CRLF UTF-8 4 spaces

I tyle…