for(initializing\_expression; loop\_condition; modifying\_expression){

loop\_body

};

String[] names = new String[5];

names[0] = "Cat";

names[1] = "Dog";

names[2] = "Horse";

names[3] = "Elephant";

names[4] = "Tiger";

for(int i=0; i<5; i++){

System.out.println("Array element names[" + i + "] = " + names[i]);

};

ArrayList<String> library = new ArrayList<String>();

library.add("50 Shades of Grey");

library.add("Sherlock Holmes");

library.add("The Last of the Mohicans");

library.add("The Infinity");

System.out.println(library.size());

for(String book: library) {

System.out.println("Element of the library is: " + book);

}

library.remove(library.size()-1);

library.add(0, "Metro 2020");

LinkedList<String> theList = new LinkedList<String>();

Wpisy Entry zapisane w mapie dzielone są na kategorie ("kubełki") na podstawie wartości obiektu Key, a nie obiektu Value.

**Porządkowanie elementów HashMapy, czyli kontrakt equals() i hashCode()**

Zastanawiasz się, jak HashMap dzieli pary zawarte w kolekcji na poszczególne kategorie ("kubełki")? Wykorzystuje do tego metodę hashCode(), obiektu reprezentującego klucz pary.

Przeanalizujmy wywołanie konstruktora mapy:

HashMap<Employee, SalaryParameters> paymentParameters = new HashMap<Employee, SalaryParameters>();

class Employee{

String firstName;

String lastName;

String peselId;

public Employee(String firstName, String lastName, String peselId){

this.firstName = firstName;

this.lastName = lastName;

this.peselId = peselId;}

public boolean equals(Object o){

final Employee e = (Employee) o;

return this.firstName.equals(e.firstName) && this.lastName.equals(e.lastName) && this.peselId.equals(e.peselId);}

public int hashCode(){

return Integer.parseInt(peselId.substring(0, 5));

}

class SalaryParameters {

Double baseSalary;

Double regulatedBonus;

Double chefBonus;

Double functionAddSalary;

public SalaryParameters(Double baseSalary, Double regulatedBonus, Double chefBonus, Double functionAddSalary){

this.baseSalary = baseSalary;

this.regulatedBonus = regulatedBonus;

this.chefBonus = chefBonus;

this.functionAddSalary = functionAddSalary;

}

public String toString(){

return baseSalary + " + " + regulatedBonus + " + " +chefBonus + " + " + functionAddSalary;

}

class HashMapTest{

public static void main (String[] args) throws java.lang.Exception{

Employee worker1 = new Employee("Jennifer", "Nowak", "12345678901");

Employee worker2 = new Employee("Sarah", "Taylor", "12345612345");

Employee worker3 = new Employee("Laura", "Pinelli", "12345698765");

Employee worker4 = new Employee("Laura", "Pinelli", "12345045678");

SalaryParameters params1 = new SalaryParameters(5000.0, 3000.0, 0.0, 0.0);

SalaryParameters params2 = new SalaryParameters(4000.0, 2000.0, 1000.0, 0.0);

SalaryParameters params3 = new SalaryParameters(4500.0, 3500.0, 0.0, 0.0);

SalaryParameters params4 = new SalaryParameters(11000.0, 0.0, 2000.0, 1000.0);

HashMap<Employee, SalaryParameters> paymentParameters = new HashMap<Employee, SalaryParameters>();

paymentParameters.put(worker1, params1);

paymentParameters.put(worker2, params2);

paymentParameters.put(worker3, params3);

paymentParameters.put(worker4, params4);

System.out.println("Salary of Jennifer Nowak equals: " + paymentParameters.get(worker1));

Sprawdźmy teraz, ile czasu zajmuje usunięcie ostatniego elementu kolekcji. Aby to zrealizować, skorzystamy z metody currentTimeMillis(), **System.nanoTime()**, biblioteki System. Dany moment w czasie możemy zapamiętać w zmiennej w następujący sposób:

long momentOfTime = System.currentTimeMillis();

//Deleting last element from the collection

long begin = System.currentTimeMillis();

list.remove(list.size()-1);

long end = System.currentTimeMillis();

System.out.println("Removing last element has taken: " + (end - begin) + "ms");

//Deleting first element from collection

begin = System.currentTimeMillis();

list.remove(0);

end = System.currentTimeMillis();

System.out.println("Removing first element has taken: " + (end - begin) + "ms");

//Adding element at the end of the collection

long begin = System.nanoTime();

list.add(list.size());

long end = System.nanoTime();

//Displaying time of the operation

System.out.println("Adding element at the od of the collection has taken: " + (end - begin) + "ns");

//Adding element at the beginning of the collection

begin = System.nanoTime();

list.add(0, list.size());

end = System.nanoTime()

//Displaying time of the operation

System.out.println("Adding element at the beginning of the collection has taken: " + (end - begin) + "ns");

public boolean equals(Object o){

Employee e = (Employee) o;

return this.firstName.equals(e.firstName) && this.lastName.equals(e.lastName) && this.peselId.equals(e.peselId);

}

public boolean equals(Object o){

final Student e = (Student) o;

return this.firstName.equals(e.firstName) && this.lastName.equals(e.lastName);

}

public boolean equals(Object o){

Employee e = (Employee) o;

return (firstName.equals(e.getFirstName())) &&

(lastName.equals(e.getLastName())) &&

(birthDate.getYear() == e.getBirthDate().getYear()) &&

(birthDate.getMonthValue() == e.getBirthDate().getMonthValue()) &&

(birthDate.getDayOfMonth() == e.getBirthDate().getDayOfMonth());

}

public boolean equals(Object o){

Book e = (Book) o;

return this.title.equals(e.title) && this.author.equals(e.author) && this.year.equals(e.year);

}

public int hashCode(){

//This method cuts first 6 (0-5) characters from peselId

//and then converts it into Integer

//peselId includes year and month and day of birth in the first 6 characters

//for example: beginning of peselId 800630 means that persone having this peselId

//was born on 30 july 1980.

//returning value will be 800630 (Integer type)

return Integer.parseInt(peselId.substring(0, 5));

}

public int hashCode(){

// methods getYear() and getMonthValue() of LocalDate class could be used

// to retrieve a year and month of stored date

return birthDate.getYear() \* 100 + birthDate.getMonthValue();

}

public int hashCode(){

return year.getYear() \* 100 + year.getMonthValue();

}