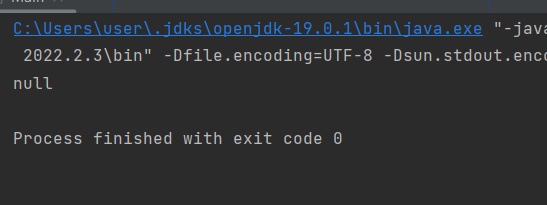
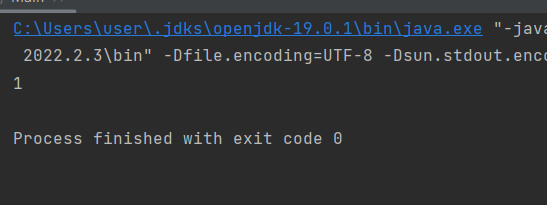
**HashMap**

* **pentru ca HashMap sa se asigure ca are key unice, mereu foloseste metoda equals() si hashCode() a keylor date. Deci, nu pur si simplu compara referintele si atat.**
* metoda **put(key,value)** daca a inlocuit un element care tot are aceeasi key, va returna acest element, dupa ce il va inlocui. Daca nu inlocuieste nici unul, returneaza null
* Atunci cand hashMap returneaza un obiect dupa key, el va folosi metoda hashCode si equals a fiecarei key pentru a vedea daca corespunde cu cel dat de noi. Totusi, deoarece fiecare obiect extinde Object, el deja ofera metoda hashCode si equals, dar sunt foarte slab implementate. Equals doar compara referintele si hashCode adresele de memorie le ia. Iata de ce, urmatorul program nu va merge cum e de asteptat:
* public class Main {  
   public static void main(String[] args) {  
    
   HashMap<Student,Integer> hashMap = new HashMap<>();  
   hashMap.put(new Student(1,"Mititiuc","Eduard"),1);  
   hashMap.put(new Student(2,"Goncear","Alexandru"),2);  
    
   Student student = new Student(1,"Mititiuc","Eduard");  
    
   System.*out*.println(hashMap.get(student));  
   }  
  }  
  class Student{  
   private int id;  
   private String firstName;  
   private String lastName;  
   public int getId() {  
   return id;  
   }  
   public Student(int id, String firstName, String lastName) {  
   this.id = id;  
   this.firstName = firstName;  
   this.lastName = lastName;  
   }  
   public void setId(int id) {  
   this.id = id;  
   }  
   public String getFirstName() {  
   return firstName;  
   }  
   public void setFirstName(String firstName) {  
   this.firstName = firstName;  
   }  
   public String getLastName() {  
   return lastName;  
   }  
   public void setLastName(String lastName) {  
   this.lastName = lastName;  
   }  
  }



desi am oferit ca key un Student ce deja exista in HashMap, dupa fileduri, HashMap nu a putut sa il gaseasca din cauza ca nu are metodele equals() si hashCode() necesare pentru a putea afla daca obiectele au fileduri identice.

* Intelij deja ofera posibilitatea de a genera aceste metode cum trebuie, la Generate alegem hashCode and equals si el le va genera:
* @Override  
  public boolean equals(Object o) {  
   if (this == o) return true;  
   if (o == null || getClass() != o.getClass()) return false;  
   Student student = (Student) o;  
   return id == student.id && firstName.equals(student.firstName) && lastName.equals(student.lastName);  
  }  
    
  @Override  
  public int hashCode() {  
   return Objects.*hash*(id, firstName, lastName);  
  }



* **Deci HashMap mereu foloseste metodele hashCode si equals ca sa caute un key, si Set tot asa fac.**

**ListIterator**

* Este aplicabil doar pentru List
* El ofera un bi directional iterator
* Este creat prin apelarea metodei **listIterator**() din orice clasa ce implementeaza List
* Methods:
* next()
* hasNext()
* nextIndex() – arata ce index va avea urmatorul element chemat cu next(). Nu trece la urmatorul obiect pana nu apela next()
* hasPrevious() – returneaza true daca itemul la care acum e iteratorul mai are unul in spatele sau
* previous() – returneaza elementul care este in spatele celui in care este iteratorul setat
* previousIndex()
* set(Object) – seteaza in lista
* add(Object) – adauga in lista in fata la itemul ce va fi returnat de next()

**next() – si la Iterator si la ListIterator ea mereu returneaza obiectul de la indexul curent si muta iteratorul pe urmatorul index din colectie. Cand iteratorul abea e returnat, el nu e setat pe nici-un item inca.**