Laborator 8

1. Utilizarea containerelor fara generics (containerele de baza)

**import** java.util.\*;

**class** Apple {

**private** **static** **long** *counter*;

**private** **final** **long** id = *counter*++;

**public** **long** id() { **return** id; }

}

**class** Orange {}

**public** **class** ApplesAndOrangesWithoutGenerics {

@SuppressWarnings("unchecked")

**public** **static** **void** main(String[] args) {

ArrayList apples = **new** ArrayList();

**for**(**int** i = 0; i < 3; i++)

apples.add(**new** Apple());

// Not prevented from adding an Orange to apples:

apples.add(**new** Orange());

**for**(**int** i = 0; i < apples.size(); i++)

((Apple)apples.get(i)).id();

// Orange is detected only at run time

}

}

1. Utilizarea containerelor generice

**import** java.util.\*;

**public** **class** ApplesAndOrangesWithGenerics {

**public** **static** **void** main(String[] args) {

ArrayList<Apple> apples = **new** ArrayList<Apple>();

**for**(**int** i = 0; i < 3; i++)

apples.add(**new** Apple());

// Compile-time error:

// apples.add(new Orange());

**for**(**int** i = 0; i < apples.size(); i++)

System.*out*.println(apples.get(i).id());

// Using foreach:

**for**(Apple c : apples)

System.*out*.println(c.id());

}

}

1. Sa se realizeze o clasa Gerbil cu o data membru de tip int denumita gerbilNumber, initializata in constructor. Clasa are o metoda hop care afiseaza data membru gerbilNumber. Create un ArrayList si adaugati obiecte de tip Gerbil in acesta. Utilizati metoda get pentru a regais obiectele , si pentru fiecare obiect apelati metoda hop. Faceti astfel incat gerbilNumber sa fie initializat cu indexul obiectului care numara cate obiecte au fost create din clasa Gerbil.
2. Inserarea in container a mai multor elemente simulatan

**import** java.util.\*;

**public** **class** AddingGroups {

**public** **static** **void** main(String[] args) {

Collection<Integer> collection =**new** ArrayList<Integer>(Arrays.*asList*(1, 2, 3, 4, 5));

Integer[] moreInts = { 6, 7, 8, 9, 10 };

collection.addAll(Arrays.*asList*(moreInts));

// Runs significantly faster, but you can’t

// construct a Collection this way:

Collections.*addAll*(collection, 11, 12, 13, 14, 15);

Collections.*addAll*(collection, moreInts);

// Produces a list "backed by" an array:

List<Integer> list = Arrays.*asList*(16, 17, 18, 19, 20);

list.set(1, 99); // OK -- modify an element

// list.add(21); // Runtime error because the

// underlying array cannot be resized.

}

}

1. Iterarea elementelor din containere

**import** java.util.\*;

**public** **class** SimpleIteration {

**public** **static** **void** main(String[] args) {

List<Apple> apples = **new** ArrayList<Apple>();

**for**(**int** i = 0; i < 3; i++)

apples.add(**new** Apple());

Iterator<Apple> it = apples.iterator();

**while**(it.hasNext()) {

Apple p = it.next();

System.*out*.print(p.id() + ":" + p + " ");

}

System.*out*.println();

// A simpler approach, when possible:

**for**(Apple p : apples)

System.*out*.print(p.id() + ":" + p + " ");

System.*out*.println();

// An Iterator can also remove elements:

it = apples.iterator();

**for**(**int** i = 0; i < 6; i++) {

it.next();

it.remove();

}

System.*out*.println(apples);

}

}