**KOLEKCJE**:

* **INTERFEJSY** (ABSTRAKCYJNE TYPY DANYCH REPREZENTUJĄCE KOLEKKCJE) (LIST,SET,MAP)
* **IMPLEMENTACJE INTERFEJSOW** (NP. ARRAYLIST, HASHMAP, HASHSET).
* **ALGORYTMY** (METODY REALIZACJI OPERACJI, POLIMORFICZNE DZIAŁAJĄCE DLA RÓŻNYCH IMPLEMENTACJI (ROZSZERZANIE KLAS

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **LIST** | | **SET** | | | **MAP** | | |
|  | **ArrayList** | **LinkedLint** | **HashSet** | **LinkedHashSet** | **TreeSet** | **HashMap** | **LinkedHashMap** | **TreeMap** |
| Kolejność elementów | TAK | | NIE | TAK | TAK | NIE | TAK | TAK |
| Powtarzalność obiektów | TAK | | NIE | | | TAK | TAK | TAK |
| Dodanie elementu | .add(E e) na koniec listy  .set(E e, int index) nadpis element z pozycji  .add(E e) lista.addAll(List.of(z1,z2,z3)); | | .add(E e)  figuraTreeSet.addAll(List.of(f1,f2,f3,f4,f5,f6)); | | | put(Key ket, Value value) –umieszcza nową parę klucz-wartość w mapie  hashMap.putAll(Map.of("imie4",3)); | | |
| Charakterystyczne | Najczęściej używana. | Szybsza manipulacja danymi. Implementuje również interfejs Queue | Nie ma bezpośredniego dostępu do obiektu poprzez indeks. Brak duplikatów. Aby dostosować się musimy skorzystać z płetli foreach lub typu **Iterator** | | | Klucz->wartość. Unikatowe klucze, dostęp po kluczu | |  |
| Dobra wydajnościowo. | - | Umieszcza nowe elementy za pomocą Comparatora | Dobra wydajnościowo | - | Umieszcza nowe elementy za pomocą Comparatora |
| Składnia | List<String> arraylist=new ArrayList<>() | List<String> linkedList=new LinkedList<>() | Set<Figura> figuraHashSet = new HashSet<>(); | Set<Figura> figuraLinkedHashSet = new LinkedHashSet<>(); | Set<Figura> figuraTreeSet = new TreeSet<>(); | HashMap<String, Integer> hashMap = new HashMap<String, Integer>(); | LinkedHashMap<String, Integer> linkedMap = new LinkedHashMap<>(); | TreeMap<String, Integer> treeMap = new TreeMap<>(); |

Odwołując się do kolekcji powinniśmy preferować użycie interfejsu zamiast konkretnego typu.