```
import Map "mo:base/HashMap"; // anahtar - kasa ilişkisi
import Hash "mo:base/Hash";
import Nat "mo:base/Nat"; // değişken şekli
import Iter "mo:base/Iter"; // iteration
import Text "mo:base/Text"; //string

// syntax'ta ; koymayı unutma !

// Define Actor
actor Assistant {
 // Akıllı sözleşmenin key'i |  canister

 type ToDo = {
   // todo listemizin içeriğini tanımlıyoruz
   description : Text;
   completed : Bool;
 };

 // fonks default olarak private gelir
 // private func nathash denebilir. : ' tan sonrası dönüş
tipini belirtir
 func natHash(n : Nat) : Hash.Hash {
   Text.hash(Nat.toText(n)) // return
Text.hash(Nat.toText(n)); ile aynı şey
 };

 var todos = Map.HashMap<Nat, ToDo>(0, Nat.equal, natHash); //
let -> immutable, var -> mutable
 var nextId : Nat = 0;

 public query func getTodos() : async [ToDo] {
   Iter.toArray(todos.vals());
 };
```

```
 public func addTodo(description : Text) : async Nat {
   let id = nextId;
   todos.put(id, { description = description; completed =
false });
   nextId += 1;
   id // return id;
 };

 // ignore do => yapılanları görmezden gel yani yapılıp
yapılmamış mı şeklinde. ?
 //işareti return ne olursa olsun yani exception mantığı
 public func completeTodo(id : Nat) : async () {
   ignore do ? {
     let description = todos.get(id)!.description;
     todos.put(id, { description; completed = true });
   };
 };

 public query func showTodos() : async Text {
   var output : Text = "\n___TO-DOs___";
   for (todo : ToDo in todos.vals()) {
     output #= "\n" # todo.description;
     if (todo.completed) { output #= " ✔" };
   };
   output # "\n";
 };

 // biten projeleri todos içerisinden kaldırıyor
 public func clearCompleted() : async () {
   todos := Map.mapFilter<Nat, ToDo, ToDo>(
     todos,
     Nat.equal,
     natHash,
     func(_, todo) { if (todo.completed) null else ?todo },
```

```
    );
};

//!!!! kendi eklediğim eksik todo crud işlemleri !!!!

//todo size'ını getir
public func getTodosSize() : async Nat {
    todos.size();
};

//todo tamamlanmayan ögeleri sil
public func nonCompletedClear() : async () {
    todos := Map.mapFilter<Nat, ToDo, ToDo>(
        todos,
        Nat.equal,
        natHash,
        func(_, todo) { if (todo.completed) ?todo else null },
    );
};

//todo tamamlanan ögeleri göster
public query func showCompletedTodos() : async Text {
    var output : Text = "\n___CompletedToDos___";
    for (todo : ToDo in todos.vals()) {
        if (todo.completed) {
            output #= "\n" # todo.description;
            output #= " ✔";
        };
    };
    output # "\n";
};

//todo tamamlanmayan ögeleri göster
public query func showNonCompletedTodos() : async Text {
```

```
    var output : Text = "\n___NonCompletedToDos___";
    for (todo : ToDo in todos.vals()) {
      if (todo.completed == false) {
        output #= "\n" # todo.description;
        output #= " x";
      };
    };
    output # "\n";
  };

  // update todo description by id
  public func updateTodoDescription(id : Nat, description :
Text) : async () {
    ignore do ? {
      let completed = todos.get(id)!.completed;
      todos.put(id, { description; completed });
    };
  };

  // delete todo by id
  public func deleteTodoById(id : Nat) : async () {
    ignore do ? {
      todos.remove(id);
    };
  };

};
```