Constructor HT6 p16	ENUM HT6 p56	Array: HT4 int[ ] naam; naam= new int[10]; int[ ] a = {-5, -2, -1, 0};		List<String> naam = new ArrayList<>();	DCD HT5 TESTEN HT8	* 1 0..1 1..* n n n..m 0..*
MAPPER HT7 p37		int[ ][ ] matrix;		@ParameterizedTest @ValueSource	Add size get set	List<String> benodigdheden;
DTO HT7 p38		prioriteit Van L>R		@Test	contains	ABSTRACTIE toegang beperk
Promo cast HT6 p54		Uitzondering: =(assignment)		@BeforeEach	Extend klasse	OVERERVING
SecureRandom		`x = y = z;` wordt x = (y = z);`			private	INCAPSULATIE
waarde ? a :b			Zelfde naam andre para =		@overload	POLYMORFISME

KLASSE:

```
package domein;
public class Rekenmachine {
Private Type naam = waarde // ATTRIBUTEN
    public Rekenmachine() {/*Lege constructor*/}
    //Setters en getters
     public Type naamMethode() {  } // Methodes  } //einde
```

APLLICATIE:

```
package cui;
 import java.util.Scanner;
import domein.DomeinController;
public class RekenmachineApplicatie {
        // Attributen komen hier met `private static Type Naam = waarde`
   public RekenmachineApplicatie() {  // Constructor
      dc = new DomeinController();   invoerScanner = new Scanner(System.in);    }
    public void run() {} //Hoofdprogramma
        private void methodeNaam () {}  // Methoden komen hier   } //einde
```

DOMEINCONTROLLER

```
package domein;
public class DomeinController {    private Rekenmachine basisRekenmachine;
   public DomeinController() { this.basisRekenmachine = new Rekenmachine();}
     public void stelGetalIn(double getal) { basisRekenmachine.setInvoerGetal(getal);  }
     public void stelTekenIn(char teken){ basisRekenmachine.setTeken(teken);  }
     public double geefResultaat() { return basisRekenmachine.voerBewerkingUit(); } }
```

STARTUP

```
package main;  import cui.RekenmachineApplicatie;
public class StartUp {    public static void main(String[] args) {
      RekenmachineApplicatie app = new RekenmachineApplicatie();       app.run();    }    } //einde
```

TESTEN

```
package testen;
import org.junit.jupiter.api.Test;    import static org.junit.jupiter.api.Assertions.assertEquals;
import domein.Rekening; public class RekeningTest {
@BeforeEach
void setUp() {r = new Rechthoek(EEN_CORRECTE_LENGTE, EEN_CORRECTE_BREEDTE);}
@Test
void testAddition() {    assertEquals(10, 5 + 5);  }   }
```

DTO=bleuprint voor gegevens (van KlasseMapper in persistentie)

```
package dto;
public record KlassenaamDTO(alle attributen van Klasse) {
   optie voor aanmaken van een subitem van KlassenDTO
        public static KlasseDTO maakExtentdeKlasseDTO(alle attributen van extented Klasse )
              ...}
```

MAPPERKLASSE=persistentie

```
package persistentie;
import java.util.ArrayList;  import java.util.List;  import dto.RekeningDTO;  import domein.Rekening;
import domein.SpaarRekening;  import domein.ZichtRekening;
public class RekeningMapper {
  public List<RekeningDTO> geefRekeningen()    {
      // We doen alsof we de inhoud van de tabel Rekening uit onze database ophalen
   List<RekeningDTO> lijst = new ArrayList<>();
      // Simulatie van het ophalen van gegevens (normaal gesproken zou je dit met SQL doen)
   Rekening sr1, sr2, zr1, zr2;
    sr1 = new SpaarRekening(101L, "Senne");
    sr2 = new SpaarRekening(202L, "Michiel");
    zr1 = new ZichtRekening(303L, "Kamiel", -2000);
    zr2 = new ZichtRekening(404L, "Jens", -2500);
     sr1.stortOp(1000);  sr2.stortOp(2000);  zr1.stortOp(3000); zr2.stortOp(4000);
      // Stel de aangroeiIntrest op 5% voor de spaarrekeningen
     sr1.setAangroeiIntrest(5.0);    sr2.setAangroeiIntrest(5.0);
      // Voeg de rekeningen toe als DTO's
       lijst.add(RekeningDTO.maakZichtRekeningDTO(zr2.getRekeningnummer(), zr2.getSaldo(),
zr2.getHouder(), zr2.getMaxkredietOnderNul()));
       lijst.add(RekeningDTO.maakSpaarrekeningDTO(sr2.getRekeningnummer(), sr2.getSaldo(),
sr2.getHouder(), sr2.getAangroeiIntrest()));
       lijst.add(RekeningDTO.maakSpaarrekeningDTO(sr1.getRekeningnummer(), sr1.getSaldo(),
sr1.getHouder(), sr1.getAangroeiIntrest()));
       lijst.add(RekeningDTO.maakZichtRekeningDTO(zr1.getRekeningnummer(), zr1.getSaldo(),
zr1.getHouder(), zr1.getMaxkredietOnderNul()));   return lijst;    }    }
```

UITVOERING VAN MAPPER EN DTO

```
import dto.RekeningDTO;  import persistentie.RekeningMapper;  import java.util.List;
public class Main {
   public static void main(String[] args) {
      RekeningMapper rekeningMapper = new RekeningMapper(); // Maak een nieuwe instantie van
de Mapper
      List<RekeningDTO> rekeningen = rekeningMapper.geefRekeningen();  // Verkrijg de lijst van
rekeningen als
      for (RekeningDTO rekeningDTO : rekeningen) {  // Toon de lijst van rekeningen (DTO's)
DTO's
        System.out.println("Rekening: " + rekeningDTO.rekeningnummer() +", Saldo: " +
rekeningDTO.saldo() +
              ", Houder: " + rekeningDTO.houder() + ", Soort: " + rekeningDTO.soort());  }   }   }
```