Προσχεδιασμένος & ευέλικτος προγραμματισμός

Σχεδιαστικά Πρότυπα (Design Patterns)

Είναι καλές πρακτικές για την επίλυση συχνών προβλημάτων που συναντώνται στην μηχανική λογισμικού. Είναι μια περιγραφή ή ένα πρότυπο για τον τρόπο επίλυσης ενός προβλήματος που μπορεί να χρησιμοποιηθεί σε πολλές διαφορετικές καταστάσεις.

Χρήσεις of design patterns

- Βοηθάει στην επικοινωνία μεταξύ προγραμματιστών
- Επιταχύνει τη διαδικασία ανάπτυξης
- Επαναχρησιμοποίηση του κώδικα
- Αναγνωσιμότητα του κώδικα

Κατηγορίες Design patterns

- Κατασκευαστικά (Creational) – About class instantiation
- Συμπεριφοράς (Structural) – About class and object composition
- Δομικά (Behavioral) – About class's object communication
- Αρχιτεκτονικά (Architectural) – About classes & project organization
- Συγχρονισμού (Concurrency) – About multi-threading

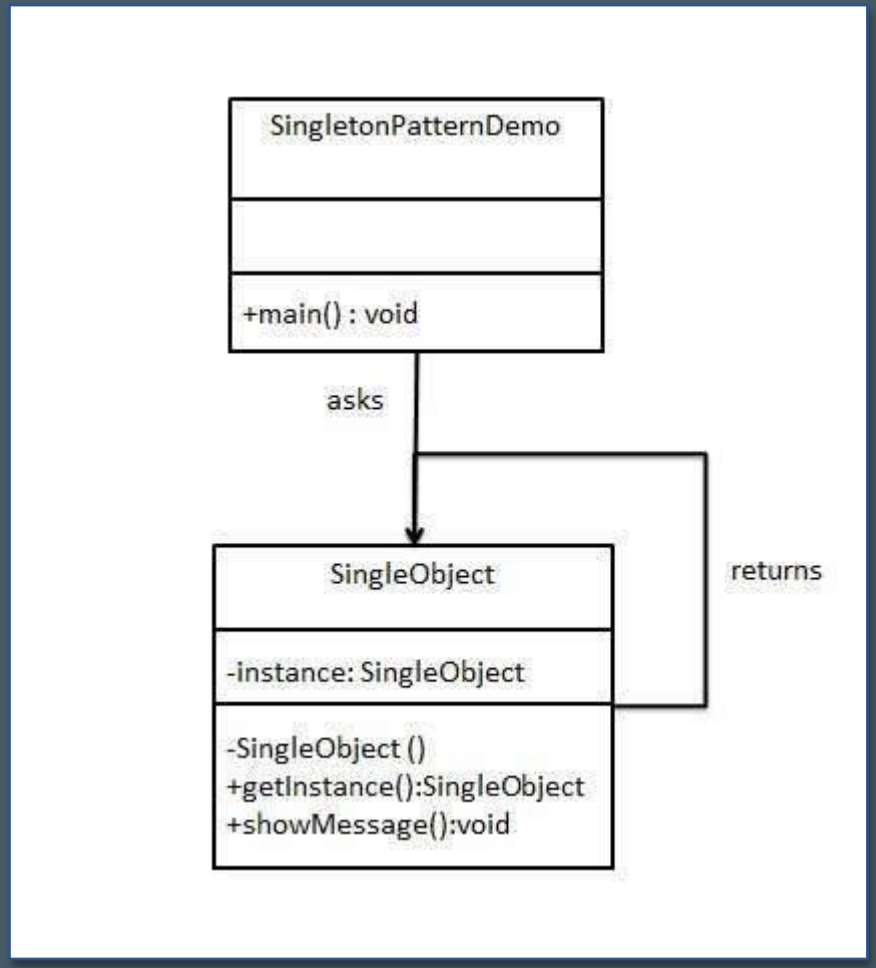
Κριτική στα design patterns

- Λείπουν επίσημα θεμέλια
- Στοχεύει στο λάθος πρόβλημα
- Οδηγεί σε αναποτελεσματικές λύσεις
- Δεν διαφέρουν σημαντικά από άλλες περιλήψεις

Σήμερα θα δούμε

- **Singleton**
- **Data Access Object (DAO)**
- **Repository**

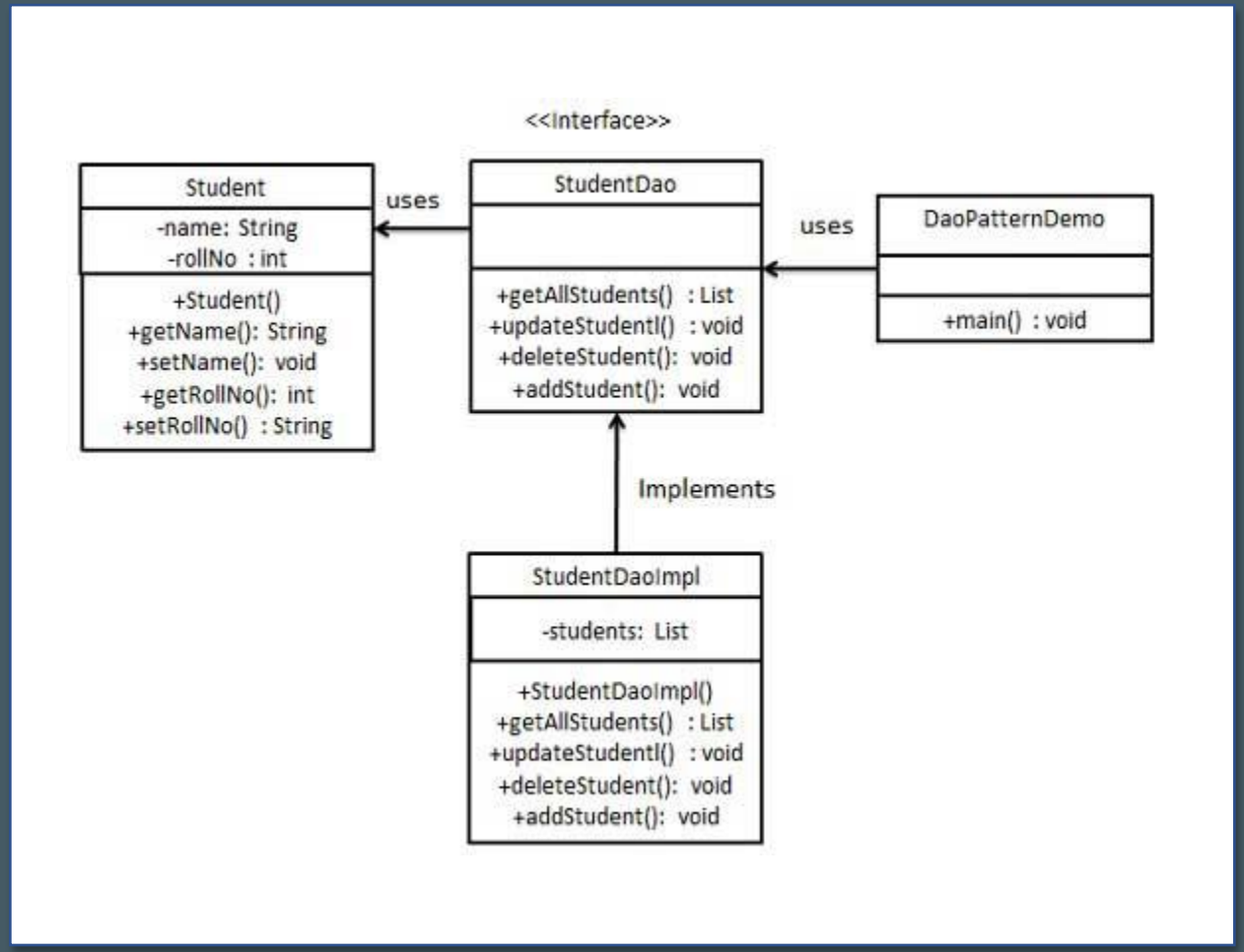
Singleton Pattern



Χρήση

- Εξασφαλίζει ότι μια κλάση έχει μόνο ένα στιγμιότυπο
- Εύκολη πρόσβαση στο μοναδικό στιγμιότυπο μιας κλάσης
- Ελέγχει την δημιουργία του
- Περιορίστε τον αριθμό των στιγμιότυπων
- Πρόσβαση σε μια καθολική μεταβλητή

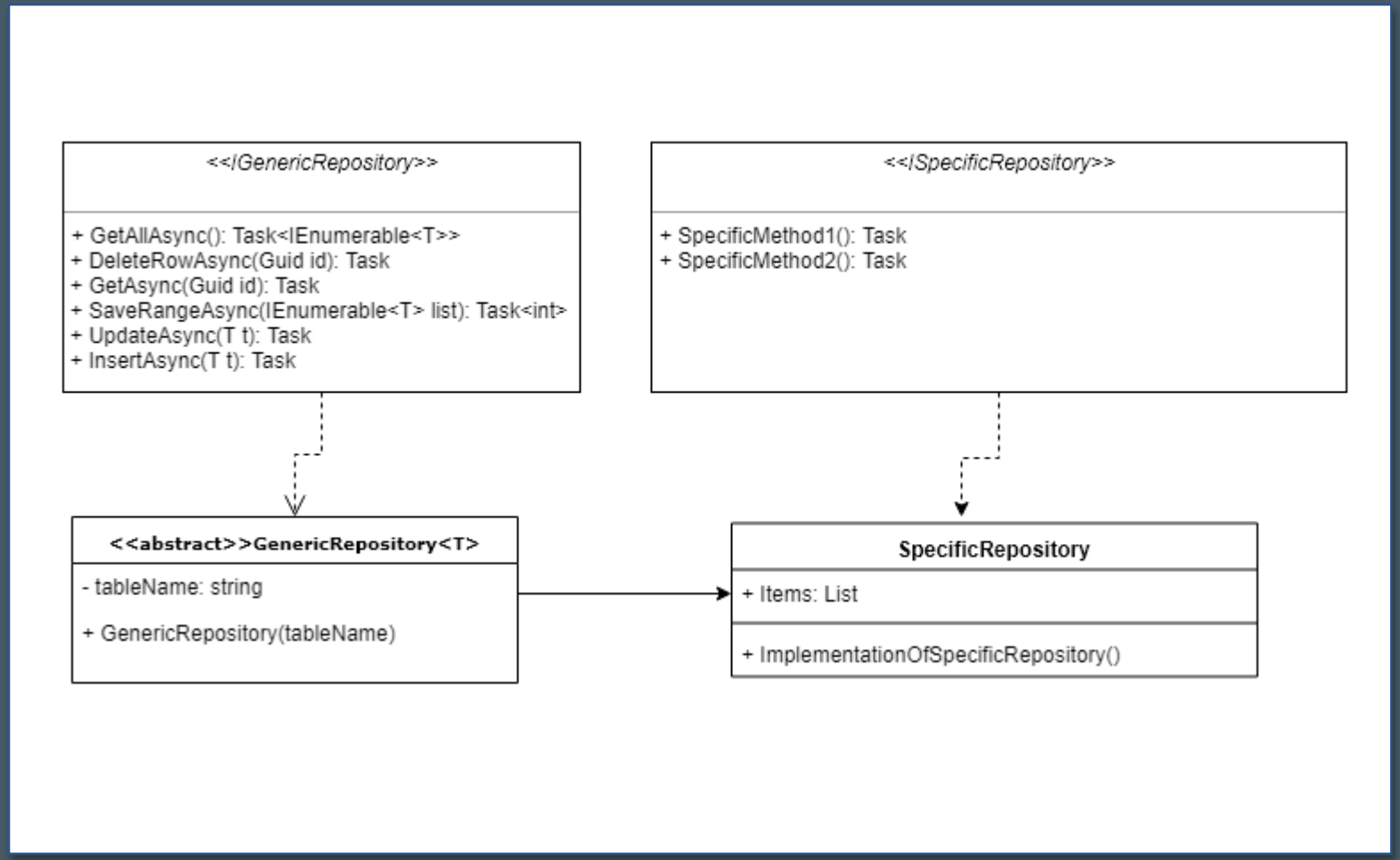
Data Access Object (DAO)



Χρήση

- Διαχωρισμός τμημάτων του κώδικα ώστε να μην γνωρίζει το ένα για το άλλο (application layer & persistence layer)
- Όλες οι λεπτομέρειες αποθήκευσης είναι κρυμμένες από την υπόλοιπη εφαρμογή
- Λειτουργεί ως ενδιάμεσος μεταξύ της εφαρμογής και της βάσης δεδομένων

Repository


<<IGenericRepository>>
+ GetAllAsync(): Task<IEnumerable<T>>
+ DeleteRowAsync(Guid id): Task
+ GetAsync(Guid id): Task
+ SaveRangeAsync(IEnumerable<T> list): Task<int>
+ UpdateAsync(T t): Task
+ InsertAsync(T t): Task
<<ISpecificRepository>>
+ SpecificMethod1(): Task
+ SpecificMethod2(): Task
<<abstract>>GenericRepository<T>
- tableName: string
+ GenericRepository(tableName)
SpecificRepository
+ Items: List
+ ImplementationOfSpecificRepository()

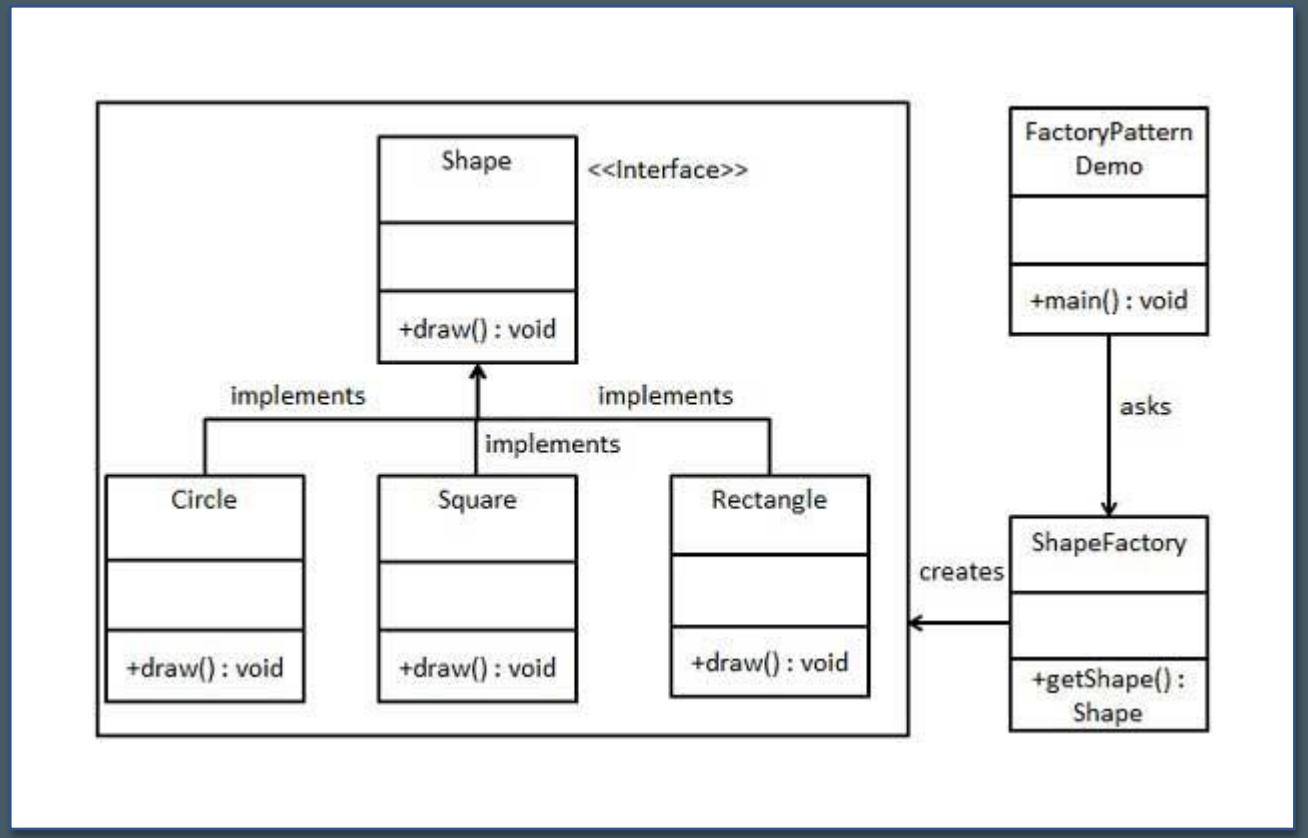
Χρήση

- Μεσολαβεί μεταξύ δύο τμημάτων (domain & data mapping layers)
- Όπως το DAO κρύβει την λειτουργικότητα κάποιων εντολών
- Προετοιμάζει τα δεδομένα να σταλούν από το domain layer στην βάση, ή πιο συνηθισμένα να φέρει δεδομένα από μια βάση ή ένα API στο domain layer.

Σήμερα θα δούμε

Factory

Factory Pattern

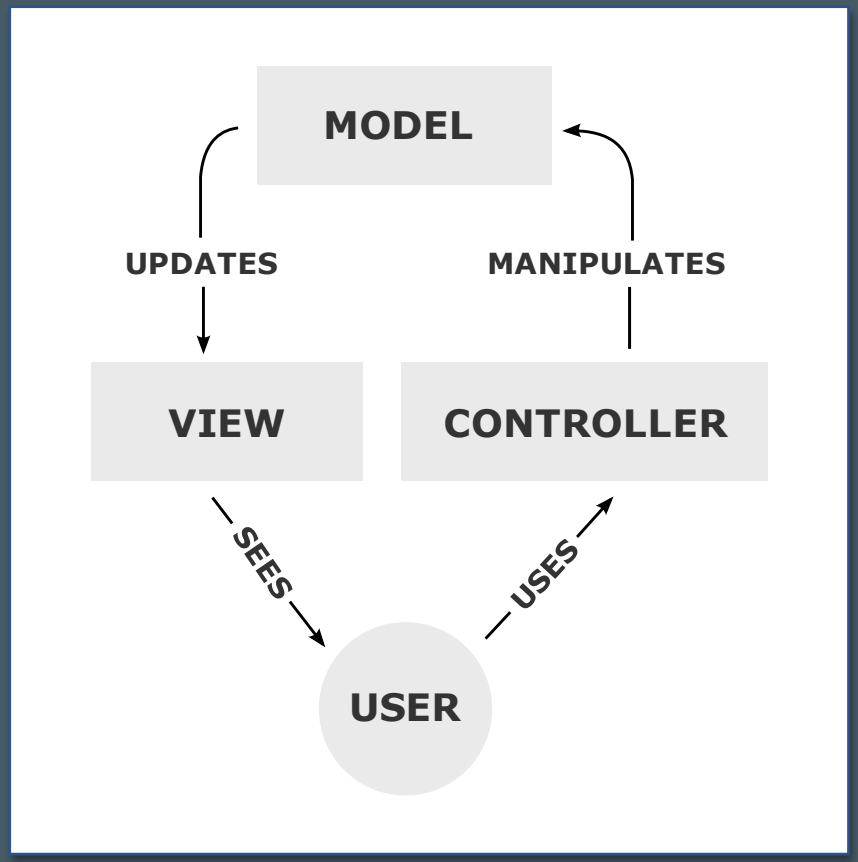


Χρήση

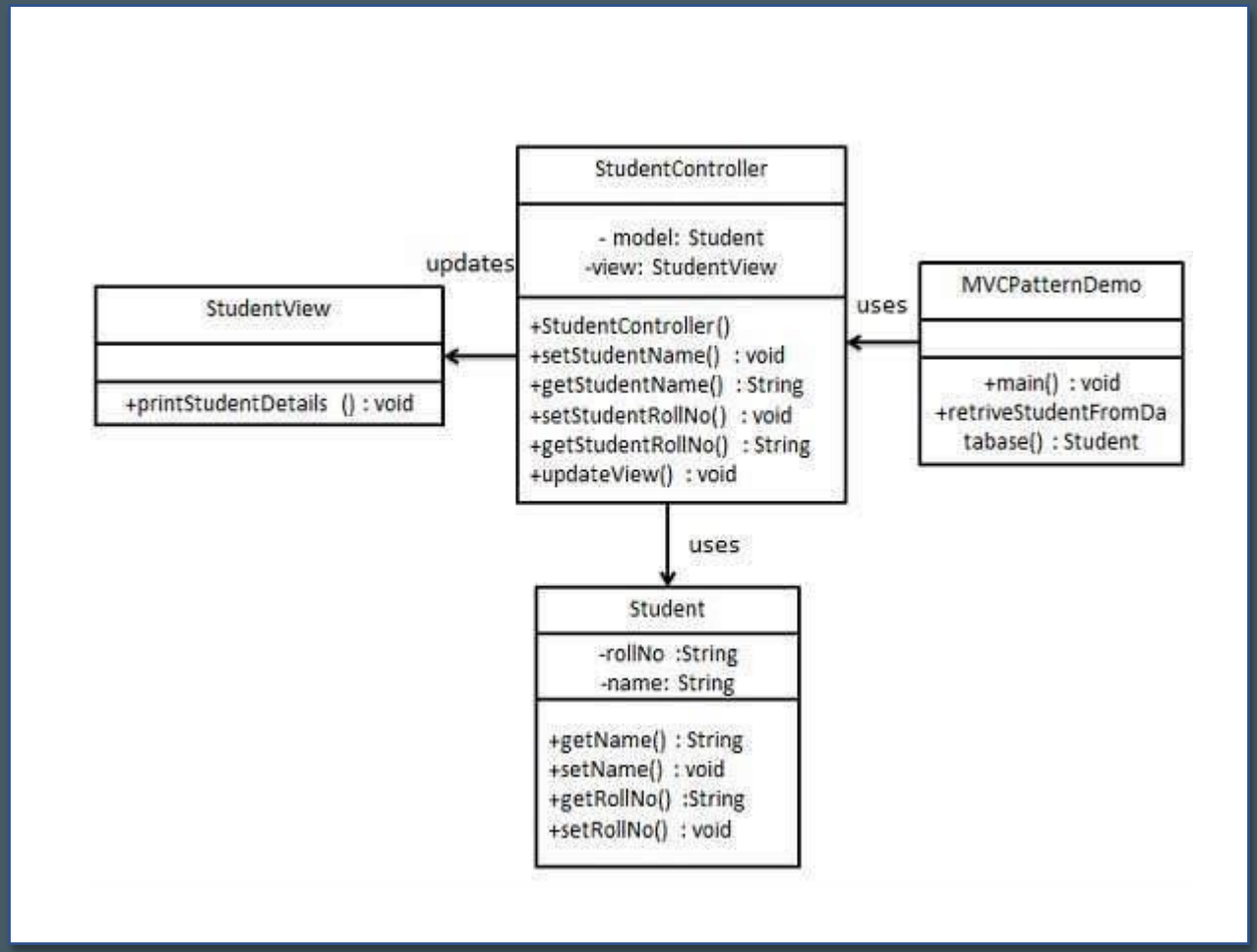
- When a class doesn't know what sub-classes will be required to create.
- When a class wants that its sub-classes specify the objects to be created.
- When the parent classes choose the creation of objects to its sub-classes.

Σήμερα θα δούμε

Model-View-Controller



MVC pattern


StudentController
- model: Student
-view: StudentView
+StudentController()
+setStudentName() : void
+getStudentName() : String
+setStudentRollNo() : void
+getStudentRollNo() : String
+updateView() : void
updates
StudentView
+printStudentDetails () : void
uses
MVCPatternDemo
+main() : void
+retriveStudentFromDatabase() : Student
uses
Student
-rollNo :String
-name: String
+getName() : String
+setName() : void
+getRollNo() :String
+setRollNo() : void

MVC

- **Model:** Περιέχει τα δεδομένα της εφαρμογής
- **View:** Περιέχει το γραφικό περιβάλλον χρήστη
- **Controller:** Περιέχει τις λειτουργίες που χειρίζονται και ελέγχουν τα δεδομένα που παρουσιάζονται

Χρήση

- Δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη
- Διαχωρίζει τις αρμοδιότητες (Separation of concerns)
- Βοηθάει στην διαχείριση του κώδικα
- Κάνει πιο εύκολες τις αλλαγές στον κώδικα

“There is an easy way and a hard way. The hard part is finding the easy way.”

Thank you