Τεχνολογία Λογισμικού

Έπανάληψη Αντικειμενοστραφή Προγραμματισμού

Τι είναι αντικείμενο?

- Στον πραγματικό κόσμο οτιδήποτε έχει μία υπόσταση.
- Κοιτάξτε γύρω σας, βρείτε αντικείμενα και προσπαθήστε να τα περιγράψετε





- Όλα έχουν καταστάσεις και συμπεριφορές.
- π.χ Το σκυλί έχει καταστάσεις (όνομα, χρώμα, ράτσα) και συμπεριφορές
- (γαυγίζει, κουνάει την ουρά).
- Στον κόσμο του OOP οτιδήποτε μπορεί να περιγραφεί σαν μια δομή και αποκτά υπόσταση στη μνήμη του συστήματος

Ανάλυση αντικειμένου

Καταστάσεις

- Τρέχουσα Ταχύτητα (20 km/h)
- Τρέχουσα Σχέση ταχυτήτων (5η)
- Τρέχων Ρυθμός πεταλιών (50 rpm)

Συμπεριφορές

- Αλλαγή σχέσης ταχυτήτων
- Αλλαγή ρυθμού πεταλιών
- Πάτημα φρένου
- Επιτάγχυνση

Διαπιστώνουμε ότι οι καταστάσεις ενός αντικειμένου μεταβάλλονται μέσω των συμπεριφορών του

Ανάλυση αντικειμένου στον προγραμματισμό

- Συνοψίζοντας κάθε αντικείμενο του πραγματικού κόσμο αναλύεται σε καταστάσεις και συμπεριφορές. Οι καταστάσεις συνήθως αλλάζουν μέσω των συμπεριφορών του αντικειμένου.

- Στον κόσμο του αντικειμενοστραφούς προγραμματισμού ένα αντικείμενο αποθηκεύει τις καταστάσεις του σε πεδία(fields) και εκφράζει τις συμπεριφορές του μέσω των μεθόδων(methods).

- Η αλλαγή τιμής μίας κατάστασης-μεταβλητής , μπορεί να γίνει κάνοντας χρήση μιας συμπεριφοράς-μεθόδου.

Fields

- Int speed;
- Int gear;
- Int cadence

Bicycle

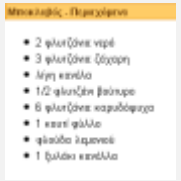
Methods

- setGear(int newValue)
- setCadence(int newValue)
- applyBreak(int decrement)
- speedup(int increment)

Τι είναι κλάση? – Περιγραφή αντικειμένων

- Η δομή με την οποία περιγράφουμε αντικείμενα-οντότητες.
  - Προσοχή, με μία κλάση ουσιαστικά περιγράφουμε αντικείμενα-οντότητες τα οποία έχουν κοινά χαρακτηριστικά.
  - Παράδειγμα, αν θέλουμε να υλοποιήσουμε την κλάση Άνθρωπος περιγράφουμε τα κοινά χαρακτηριστικά που έχουν όλοι οι άνθρωποι μεταξύ τους και τα οποία τον διαφοροποιούν από τα υπόλοιπα ζώα.
- Από μία κλάση δημιουργούμε όσα αντικείμενα επιθυμούμε.
- **Είναι το προσχέδιο(blueprint) από το οποίο δημιουργούμε ένα η περισσότερα αντικείμενα ιδίου τύπου.**
- Θα μπορούσαμε να παρομοιάσουμε την κλάση σαν:

α) Συνταγή ενός γλυκού :



β) Αρχιτεκτονικό σχέδιο:

Σύνταξη κλάσεων

```
public class Bicycle {
   int cadence;
   int speed;
   int gear;

   public Bicycle() {
   }

   public Bicycle(int cadence, int speed, int gear) {
      this.cadence = cadence;
      this.speed = speed;
      this.gear = gear;
   }

   public int getCadence() {
      return cadence;
   }


   public int getSpeed() {
      return speed;
   }

   public int getGear() {
      return gear;
   }

   public void setCadence(int cadence) {
      this.cadence = cadence;
   }

   public void setSpeed(int speed) {
      this.speed = speed;
   }


   public void setGear(int gear) {
      this.gear = gear;
   }

   public void speedUp(int increment) {
      speed = speed + increment;
   }

   public void applyBrake(int decrement) {
      speed = speed - decrement;
   }
}
```

1. Όνομα κλάσης
2. Πεδία - Καταστάσεις
3. Constructors
4. Getters
5. Setters
6. Μέθοδοι - Συμπεριφορές

Constructors

- Ο Constructor μοιάζει με μία μέθοδο, με την διαφορά ότι έχει ως όνομα το όνομα της κλάσης και δεν έχει τιμή επιστροφής.

- Ο Constructor είναι αυτός που καλείται όταν δημιουργούμε ένα αντικείμενο.

- Ο Constructor συνήθως αρχικοποιεί τα πεδία του αντικειμένου.

- Σε μία κλάση μπορούμε να έχουμε περισσότερους του ενός Constructors αρκεί να έχουν διαφορετική λίστα παραμέτρων (constructor overload).

```
public Bicycle() {
    cadence = 0;
    speed = 0;
    gear = 1;
}


public Bicycle(int cadence, int speed, int gear) {
    this.cadence = cadence;
    this.speed = speed;
    this.gear = gear;
}
```

Getters - Setters

- Getters
  - Getter είναι μία μέθοδος με την οποία διαβάζουμε την τιμή ενός πεδίου μίας κλάσης

```
public int getGear() {
   return gear;
}
```

- Setters
  - Setter είναι μία μέθοδος με την οποία αλλάζουμε την τιμή ενός πεδίου μίας κλάσης

```
public void setGear(int newValue)
{
   this.gear = newValue;
}
```

Υπερφόρτωση μεθόδων - Method Overload

- Σε μία κλάση μπορούμε να έχουμε περισσότερες από μια μεθόδους με το ίδιο όνομα αρκεί να έχουν διαφορετική λίστα παραμέτρων

```
public class Bicycle {
   int cadence;
   int speed;
   int gear;

   public void speedUp(int
increment) {
      speed = speed +
increment;
   }

   public void speedUp() {
      speed += 10;
   }

   public void speedUp(int
increment, int maxSpeed) {
     if((speed + increment) <
maxSpeed) {
        speed += increment;
     } else {
        speed = maxSpeed;
     }
   }
}
```

Κανόνες ορθής σύνταξης

- Όνομα κλάσης άρα και constructor ξεκινάει με κεφαλαίο.
- Όνομα πεδίου, μεταβλητής ή μεθόδου ξεκινάει με μικρό.
- Η πρώτη λέξη στο όνομα μεθόδου είναι ρήμα.
- Εάν ένα όνομα αποτελείται από περισσότερες από μία λέξεις, τότε κάθε επόμενη λέξη ξεκινά με κεφαλαίο.
- Στην σύνταξη κλάσεων ακολουθούμε την σειρά

1. Πεδία
2. Constructors
3. Getters
4. Setters
5. Μέθοδοι

Δημιουργία αντικειμένων

```
public static void main(String[] args) {
    Bicycle myBike = new Bicycle(20, 15,5);
    Bicycle yourBike = new Bicycle();

    System.out.println(myBike.getGear());
    System.out.println(yourBike.getGear());
}
```

Με την εκτέλεση του πρώτου μισού της εντολής (Bicycle myBike) γίνεται δέσμευση χώρου μνήμης για μια μεταβλητή τύπου Bicycle.

Με την εκτέλεση του δεύτερου μισού της εντολής (myBike = new Bicycle(20,15,5);) δημιουργείται ένα στιγμιότυπο του Bicycle και εκχωρείται στην μεταβλητή myBike η οποία είναι τύπου Bicycle.

Με την εκτέλεση της εντολής myBike.getGear(); Καλούμε την μέθοδο getGear() της κλάσης Bicycle και τυπώνουμε την τιμή που επιστρέφει

Ασκήσεις

a) Υλοποιήστε την κλάση του αντικειμένου «Αυτοκίνητο» ,έτσι ώστε κάθε κλάση θα έχει τουλάχιστον δύο καταστάσεις (fields), τέσσερις συμπεριφορές (methods), δύο κατασκευαστές και όλες τις μεθόδους Setters και Getters. Στην συνέχεια να υλοποιήσετε μια κλάση Main(η οποία θα έχει μέθοδο main()) στην οποία να δημιουργήσετε αντικείμενα των παραπάνω κλάσεων και να καλείτε όλες τις μεθόδους τους.

b) Υλοποιήστε την κλάση του αντικειμένου «Άνθρωπος» , έτσι ώστε κάθε κλάση θα έχει τουλάχιστον δύο καταστάσεις (fields), τέσσερις συμπεριφορές (methods), δύο κατασκευαστές και όλες τις μεθόδους Setters και Getters. Στην συνέχεια να υλοποιήσετε μια κλάση Main(η οποία θα έχει μέθοδο main()) στην οποία να δημιουργήσετε αντικείμενα των παραπάνω κλάσεων και να καλείτε όλες τις μεθόδους τους.

Προαιρετικές ασκήσεις

- Υλοποιήστε τις κλάσεις του αντικειμένου «Τρένο», «Ρολόι» και «Βιβλίο». Κάθε κλάση θα έχει τουλάχιστον δύο μεταβλητές (variables), δύο κατασκευαστές, όλες τις μεθόδους Setters και Getters και μία επιπλέον μέθοδο. Στην συνέχεια να υλοποιήσετε μια κλάση Main(η οποία θα έχει μέθοδο main()) στην οποία να δημιουργήσετε αντικείμενα των παραπάνω κλάσεων και να καλείτε όλες τις μεθόδους τους.

Access modifiers

- Καθορίζουν το επίπεδο προσβασιμότητας μιας μεταβλητής ή μιας μεθόδου μίας κλάσης από άλλες κλάσεις.
  - public → η μεταβλητή / μέθοδος είναι προσβάσιμη από όλες τις κλάσεις
  - private → η μεταβλητή / μέθοδος είναι προσβάσιμη μόνο μέσα στην ίδια την κλάση

```
public class Bicycle {
    public int speed;
    private int gear;

    public Bicycle() {
        speed = 0;
        gear = 1;
    }

    private int getSpeed() {
        return speed;
    }

    public int getGear() {
        return gear;
    }
}
```

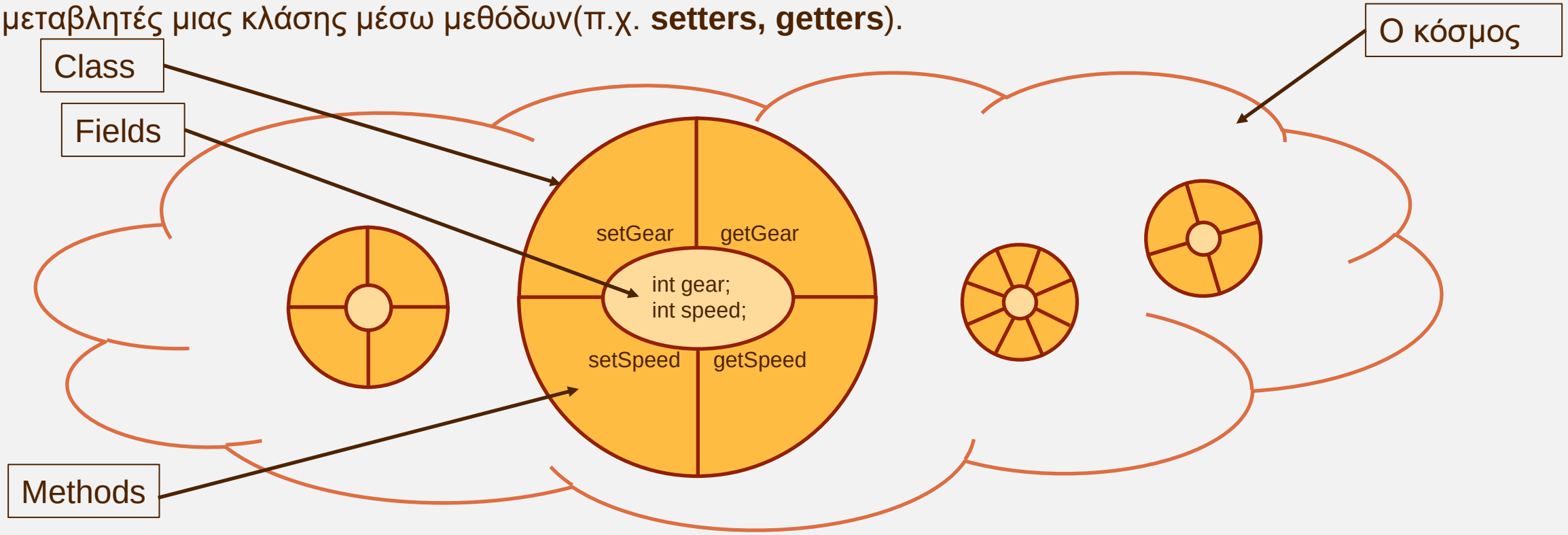
```
public class Main {
    public static void main(String[] args) {
        Bicycle myBike = new Bicycle(20, 15,5);

        System.out.println(myBike.speed);
        System.out.println(myBike.getSpeed());

        System.out.println(myBike.gear);
        System.out.println(myBike.getGear());
    }
}
```

Ενθυλάκωση (encapsulation)

- Η τεχνική κατά την οποία αποκρύπτεται η εσωτερική κατάσταση της κλάσης και η αλληλεπίδραση μαζί της γίνεται μέσω των μεθόδων.
- Δηλώνουμε τα fields ως private και κάνουμε προσπέλαση (ανάγνωση ή αλλαγή τιμής) στις μεταβλητές μιας κλάσης μέσω μεθόδων(π.χ. **setters, getters**).



Instance and Class Members (1/3)

- **Instance Variable/Method** → Κάθε αντικείμενο της κλάσης έχει το δικό του στιγμιότυπο (αντίγραφο) αυτής της μεταβλητής/μεθόδου.
- **Class Variable / Method** → Όλα τα αντικείμενα της κλάσης έχουν κοινή αυτήν την μεταβλητή/μέθοδο.

- **Για παράδειγμα:** Θέλουμε σε κάθε ποδήλατο να κρατάμε ένα μοναδικό σειριακό αριθμό. Στο πρώτο ποδήλατο αυτός θα είναι το 1 στο δεύτερο το 2 κ.ο.κ.
  - Το κάθε ποδήλατο λοιπόν έχει τον δικό του σειριακό αριθμό και γι` αυτό είναι instance variable

```
private int id;
```

  - Θα πρέπει όμως να κρατάμε κάπου τον αριθμό των ποδηλάτων που έχει δημιουργηθεί για να ξέρουμε ποιόν σειριακό αριθμό να δώσουμε στον επόμενο. Έτσι το κρατάμε σε μια class variable

```
private static int numberOfBicycles;
```

Instance and Class Members (2/3)

```
public class Bicycle {
//    instance variable
    private int id;
//    class variable
    private static int numberOfBicycles = 0;
//    class constructor
    public Bicycle() {
        id = ++numberOfBicycles;
    }
//    instance method - getter
    public int getId() {
        return id;
    }
//    class method - getter
    public static int getNumberOfBicycles() {
        return numberOfBicycles;
    }
//    instance method - setter
    public void setId(int id) {
        this.id = id;
    }
//    class method - setter
    public void setNumberOfBicycles (int numberOfBicycles) {
        Bicycle.numberOfBicycles = numberOfBicycles;
    }
}
```

Η κλάση Bicycle με instance και class members

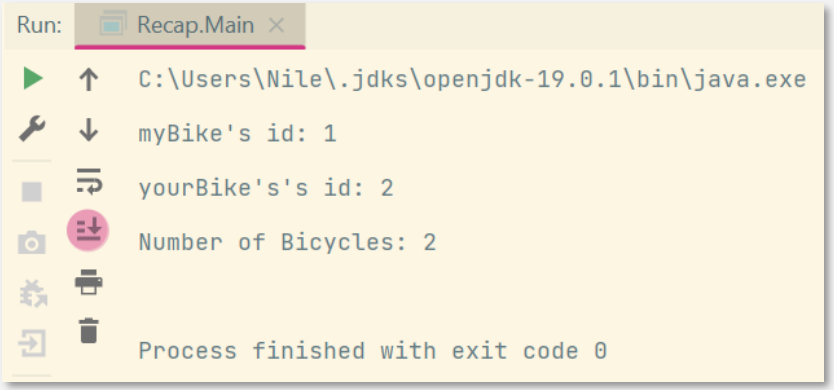
Instance and Class Members (3/3)

```
public class Main {
   public static void main(String[] args) {
      Bicycle myBike = new Bicycle(20, 15,5);
      Bicycle yourBike = new Bicycle();

      System.out.println("myBike's id: " +
myBike.getId());
      System.out.println("yourBike's's id: " +
yourBike.getId());

      System.out.println("Number of Bicycles: " +
Bicycle.getNumberOfBicycles());
   }
}
```

Η main όπου κάνουμε κλήσεις τόσο των instance methods όσο και των class methods



Το output αφού τρέξουμε την παραπάνω main

Κληρονομικότητα – Inheritance (1/3)

- Κάποιες κλάσεις αντικειμένων έχουν κοινά χαρακτηριστικά με άλλες κλάσεις
- **Κληρονομικότητα** είναι η δυνατότητα της κλάσης να κληρονομεί καταστάσεις και συμπεριφορές από μία άλλη
- Μια κλάση που κληρονομεί τα χαρακτηριστικά μια άλλης κλάσης (**υπερκλάσης**) ονομάζεται υποκλάση της κλάσης αυτής
- Κάθε κλάση περιέχει όλα τα χαρακτηριστικά της υπερκλάσης της και κάποια επιπλέον
- Κάθε κλάση μπορεί να έχει πολλές υποκλάσεις αλλά μόνο μια υπερκλάση
- Έτσι δημιουργείται η ιεραρχία της Java στην κορυφή της οποίας βρίσκεται η κλάση **Object**

Κληρονομικότητα – Inheritance (2/3)

Bicycle

Υπερκλάση των κλάσεων Mountain Bike, Road Bike και Tandem Bike

Mountain Bike

Υποκλάση της κλάσης Bicycle

Road Bike

Υποκλάση της κλάσης Bicycle

Tandem Bike

Υποκλάση της κλάσης Bicycle

Κληρονομικότητα – Inheritance (3/3)

Bicycle

```
public class Bicycle {
   public int speed;
   private int gear;

   public Bicycle() {
      speed = 0;
      gear = 1;
   }

   private int getSpeed() {
      return speed;
   }

   public int getGear() {
      return gear;
   }
}
```

extends

Mountain Bike

```
public class MountainBike extends Bicycle{
   private int seatHeight;
}
```

extends

Road Bike

```
public class RoadBike extends Bicycle{
   private int tireWidth;
}
```

extends

Tandem Bike

```
public class TandemBike extends Bicycle{
   private int numberOfSeats;
}
```

super / this keywords

- **super** → Χρησιμοποιείται μέσα σε μία κλάση και μας δίνει πρόσβαση στα πεδία, τους κατασκευαστές ή τις μεθόδους της υπερκλάσης
- **this** → Χρησιμοποιείται μέσα σε μία κλάση και μας δίνει πρόσβαση στα πεδία, τους κατασκευαστές ή τις μεθόδους της ίδια της κλάσης

```
public class MountainBike extends Bicycle{
    private int seatHeight;

    public MountainBike() {
        super();
        seatHeight = 1;
    }

    public int getSeatHeight() {
        return seatHeight;
    }

    public void setSeatHeight(int seatHeight) {
        this.seatHeight = seatHeight;
    }

    @Override
    public void printDescription() {
        super.printDescription();
        System.out.println("Seat Height: " + this.getSeatHeight());
    }
}
```

Κλήση του constructor της υπερκλάσης

Κλήση του field αυτής της κλάσης

Κλήση μεθόδου της υπερκλάσης

Πολυμορφισμός – Polymorphism

- **Πολυμορφισμός** ονομάζεται η ιδιότητα των υποκλάσεων τόσο να ορίζουν δικές τους συμπεριφορές όσο και ταυτόχρονα να διατηρούν κάποιες άλλες όπως έχουν οριστεί στις υπερκλάσεις τους.

```
public class Bicycle {
   private int speed;
   private int gear;

   public Bicycle() {
      speed = 0;
      gear = 1;
   }

   public int getSpeed() {
      return speed;
   }

   public int getGear() {
      return gear;
   }

   public void printDescription() {
      System.out.println("Gear: " + gear);
      System.out.println("Speed: " + speed);
   }
}
```

```
public class MountainBike extends Bicycle{
   private int seatHeight;

   public MountainBike() {
      super();
      seatHeight = 1;
   }

   public int getSeatHeight() {
      return seatHeight;
   }

   public void setSeatHeight(int seatHeight) {
      this.seatHeight = seatHeight;
   }

   @Override
   public void printDescription() {
      System.out.println("Gear: " + this.getGear());
      System.out.println("Speed: " + this.getSpeed());
      System.out.println("Seat Height: " + this.getSeatHeight());
   }
}
```

Η διαδικασία αυτή λέγεται “method overriding” και δίνει την δυνατότητα σε μία κλάση να διαφοροποιεί μία μέθοδο που κληρονομεί από την υπερκλάση της, διατηρώντας ίδια την υπογραφή της (όνομα μεθόδου, αριθμός και τύπος παραμέτρων)

Ασκήσεις

- Δημιουργήστε την κλάση «Άνθρωπος» και 2 υποκλάσεις του έτσι ώστε κάθε κλάση να έχει τουλάχιστον ένα κατασκευαστή, δύο μεταβλητές, όλες τις μεθόδους setters και getters και άλλη μία μέθοδο(εκτός setters-getters). Επίσης, να φαίνεται στην άσκηση ο τρόπος με τον οποίο μια κλάση μπορεί να κάνει κλήση των μεταβλητών και μεθόδων του εαυτού της και της υπερκλάσης της, με την χρήση των this και super. Στην συνέχεια να υλοποιήσετε μια κλάση Main (η οποία θα έχει μέθοδο main()) στην οποία θα δημιουργήσετε στιγμιότυπα (instances) των κλάσεων σας και θα καλείτε τις μεθόδους τους.

Προαιρετικές ασκήσεις

- Δημιουργήστε την κλάση (class) του αντικειμένου «Τηλέφωνο». Η κλάση αυτή να έχει μία υπερκλάση και δύο υποκλάσεις που θεωρείτε κατάλληλες. Όλες οι κλάσεις που θα δημιουργήσετε να έχουν τουλάχιστον ένα κατασκευαστή, δύο μεταβλητές, όλες τις μεθόδους setters και getters και άλλη μία μέθοδο(εκτός setters-getters). Επίσης, να φαίνεται στην άσκηση ο τρόπος με τον οποίο μια κλάση μπορεί να κάνει κλήση των μεταβλητών και μεθόδων του εαυτού της και της υπερκλάσης της, με την χρήση των this και super. Στην συνέχεια να υλοποιήσετε μια κλάση Main (η οποία θα έχει μέθοδο main()) στην οποία θα δημιουργήσετε στιγμιότυπα (instances) των κλάσεων σας και θα καλείτε τις μεθόδους τους.

Interfaces (1/3)

- Ένα interface είναι ένα «συμβόλαιο» που ορίζει σε μία κλάση αντικειμένων το πώς θα αλληλεπιδράει με άλλες κλάσεις και ον υπόλοιπο κόσμο
  - Αποτελεί μια δομή η οποία δηλώνεται με το keyword interface και περιέχει μία **συλλογή από μη υλοποιημένες μεθόδους**.
  - Οι μέθοδοι αυτοί ορίζουν στην ουσία τις συμπεριφορές που θα πρέπει οπωσδήποτε να έχει ένα αντικείμενο που «υλοποιεί» το συγκεκριμένο interface.
  - Ορίζεται ως μέσο προτυποποίησης (standardization) του τρόπου επικοινωνίας διαφορετικών τύπων αντικειμένων.
  - Αποτελεί μια κατευθυντήρια γραμμή (guideline) που υποβοηθά τον προγραμματιστή να αναπτύξη μια κλάση.

Interfaces (2/3)

```
public interface Vehicle {
   void speedUp(int increment);
   void applyBreak(int decrement);
}
```

Δηλώνουμε το όνομα του interface και μέσα δηλώνουμε τις μεθόδους που θα πρέπει να υλοποιεί κάθε κλάση που κάνει implement αυτό το interface!

```
public class Bicycle implements Vehicle {
private int speed;
   private int gear;

   public Bicycle() {

      speed = 0;
      gear = 1;
   }

   public void speedUp(int increment) {
      speed = speed + increment;
   }

   public void applyBrake(int decrement) {
      speed = speed - decrement;
   }
}
```

Χρησιμοποιούμε το keyword implements για να δηλώσουμε ότι η κλάση αυτή θα υλοποιήσει το interface Vehicle.

Η κλάση θα πρέπει οπωσδήποτε τώρα να περιέχει υλοποίηση των μεθόδων που τις έχει ορίσει το interface!

Interfaces (3/3)

- Επειδή οι μέθοδοι ενός interface είναι πάντα public, θα πρέπει να δηλωθούν ως public όταν υλοποιηθούν μέσα σε μία κλάση
- Τα interface έχουν σταθερές μεταβλητές και μεθόδους χωρίς σώμα
- Ένα interface μπορεί να κάνει extends άλλα (ένα ή περισσότερα) interfaces και να κληρονομήσει τις μεθόδους του
- Μια κλάση μπορεί να κάνει implements πολλά interfaces (αλλά extends μόνο μία κλάση)
- Κάθε κλάση που κάνει implements ένα interface είναι αναγκασμένη να υλοποιήσει όλες τις μεθόδους του.

Αφηρημένες Κλάσεις – Μέθοδοι (1/3)

- Με το keyword abstract ορίζουμε αφηρημένες κλάσεις και μεθόδους
- Μια αφηρημένη κλάση μπορεί να έχει ή όχι αφηρημένες μεθόδους.
- Οι αφηρημένες κλάσεις ΔΕΝ μπορούν να δημιουργηθούν, μπορούν όμως να κληρονομηθούν. Γι` αυτό το λόγο μπορεί να διαθέτει constructors
- Εάν μία κλάση περιέχει αφηρημένες μεθόδους, τότε η κλάση πρέπει να δηλωθεί αφηρημένη
- Όταν μια αφηρημένη κλάση κληρονομηθεί, τότε η υποκλάση της είναι υποχρεωμένη να υλοποιήσει όλες τις αφηρημένες μεθόδους της. Αν δεν το κάνει τότε θα πρέπει και η υποκλάση της να δηλωθεί ως abstract κ.ο.κ.
- Ιδιότητες όπως method override κτλ παραμένουν στης αφηρημένες κλάσεις
- Μια αφηρημένη μέθοδος δηλώνεται χωρίς σώμα και χωρίς αγκύλες

Αφηρημένες Κλάσεις – Μέθοδοι (2/3)

```
public class Bicycle implements Vehicle {
   private int speed;
   private int gear;

   public Bicycle() {
      speed = 0;
      gear = 1;
   }

   public int getSpeed() {
      return speed;
   }
   public int getGear() {
      return gear;
   }

   public void setSpeed(int speed) {
      this.speed = speed;
   }
   public void setGear(int gear) {
      this.gear = gear;
   }

   public abstract void speedUp(int increment);
   public abstract void applyBrake(int decrement);
}
```

```
public interface Vehicle {
   void speedUp(int increment);
   void applyBreak(int decrement);
}
```

Αφού είναι abstract κλάση δεν μπορεί να δημιουγηθεί κάποιο instance της. Δηλαδή δεν μπορεί να καλεστεί ο constructor της μετά το new στην main

```
public class Main {
   public static void main(String[] args) {
      Bicycle myBike = new Bicycle();
   }
}
```

Η κλάση Bicycle έχει κάποιες υλοποιημένες μεθόδους αλλά έχει και κάποιες abstract και γι`αυτό δηλώνεται ως abstract. Αυτές οι μεθόδοι θα πρέπει να υλοποιηθούν από την υποκλάση της

Αφηρημένες Κλάσεις – Μέθοδοι (3/3)

```
public class MountainBike extends Bicycle{
    private int seatHeight;

    public MountainBike() {
        super();
        seatHeight = 1;
    }

    public int getSeatHeight() {
        return seatHeight;
    }

    public void setSeatHeight(int seatHeight) {
        this.seatHeight = seatHeight;
    }

    @Override
    public void applyBrake(int decrement) {
        super.applyBrake(decrement);
    }

    @Override
    public void speedUp() {
        super.speedUp();
    }
}
```

Παρόλο που η υπερκλάση της είναι abstract (δεν μπορούν να δημιουργηθούν instance της) ο constructor της μπορεί να καλεστεί μέσω αυτής της υποκλάσης

Η υποκλάση MountainBike της abstract class Bicycle υλοποιεί όλες τις abstract μεθόδους της.

Διαφορές μεταξύ Interface και Abstract Class

- Οι αφηρημένες κλάσεις μπορούν να περιέχουν και fields που δεν είναι σταθερές (static – final) ενώ τα Interface όχι

- Οι αφηρημένες κλάσεις μπορούν να περιέχουν υλοποιημένες μεθόδους ενώ τα Interface δεν μπορούν. Αν μια αφηρημένη κλάση περιέχει μόνο abstract methods θα έπρεπε να δηλωθεί σαν Interface

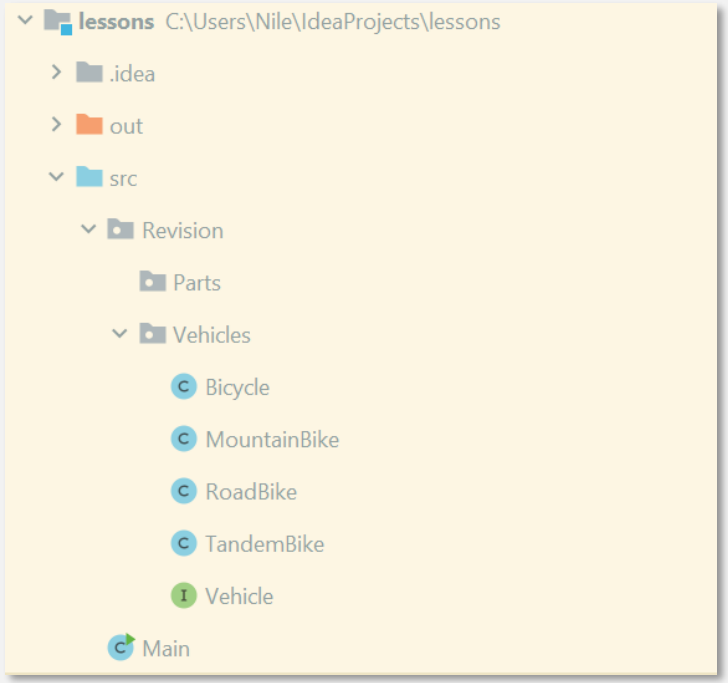
- Οι αφηρημένες κλάσεις περιέχουν μέρος της υλοποίησης και αφήνουν στις υποκλάσεις την υπόλοιπη. Τα Interfaces δεν έχουν καθόλου υλοποίηση παρά μόνο δηλώσεις

- Τα Interfaces υλοποιούνται οπουδήποτε στην ιεραρχία κλάσεων ενώ οι abstract classes όχι

- Μια αφηρημένη κλάση χωρίζεται συνήθως σε πολλές παρόμοιες υποκλάσειςπου έχουν πολλά κοινά στοιχεία (υλοποιημένες μέθοδοι) αλλά και κάποιες διαφορές (αφηρημένες μέθοδοι)

Πακέτα - Packages

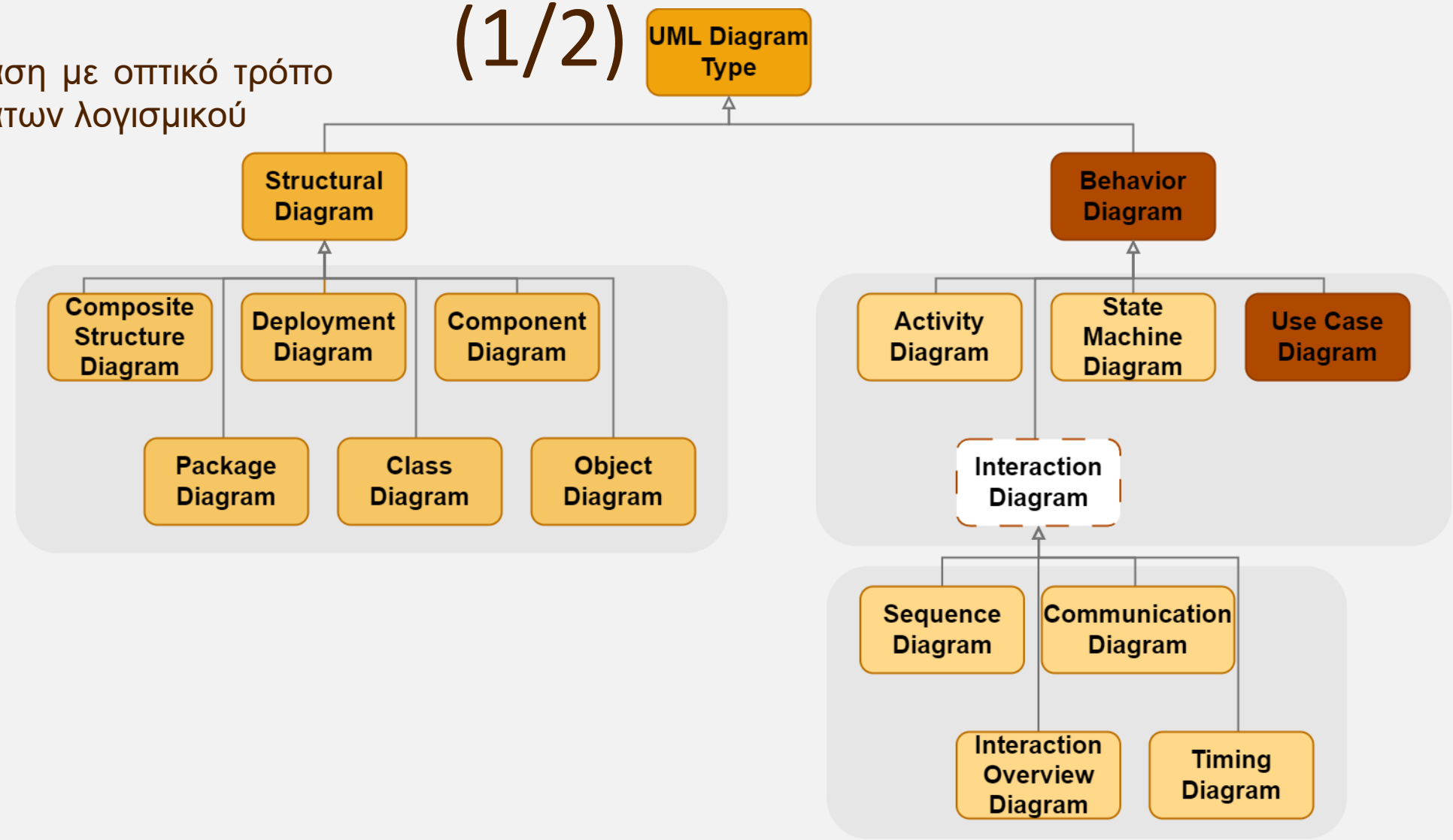
- Είναι μια δομή που μας βοηθάει να ομαδοποιήσουμε κλάσεις και interfaces. Κάτι σαν τους φάκελους στον υπολογιστή μας



- Το Java Platform API Specification μας δίνει μια λίστα από όλα τα υλοποιημένα packages, interfaces, classes, fields, και methods που μας παρέχονται έτοιμα από την Java.

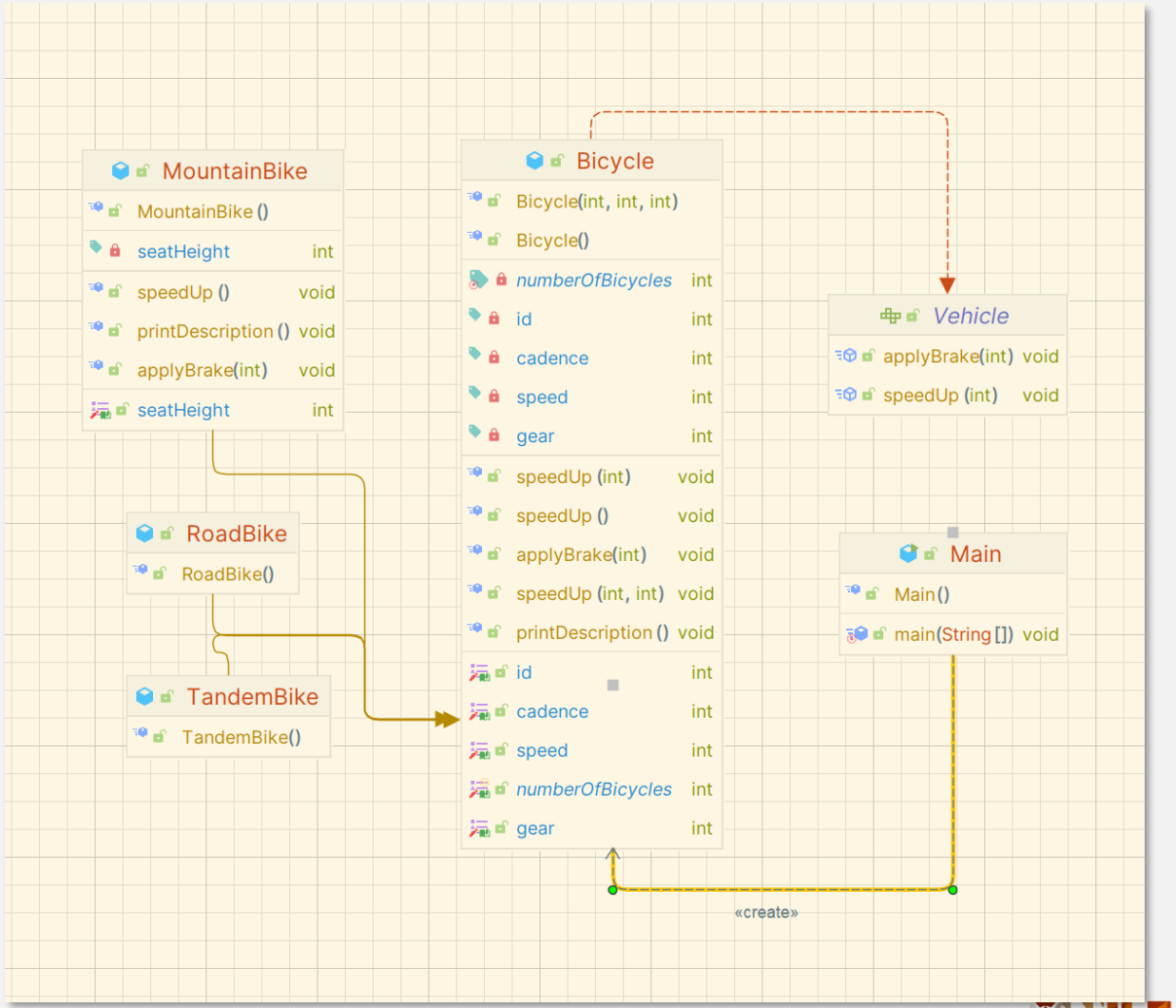
UML (Unified Modeling Language) – Class Diagrams (1/2)

- **UML** - Αναπαράσταση με οπτικό τρόπο (visualization) τμημάτων λογισμικού



UML (Unified Modeling Language) – Class Diagrams (2/2)

- **Class Diagrams** – Περιγράφουν την δομή ενός συστήματος με το να απεικονίζουν τις κλάσεις, τα πεδία και τις μεθόδους τους καθώς και τις σχέσεις μεταξύ των διάφορων κλάσεων



Ασκήσεις

- Δημιουργήστε το interface «vehicle», τις υπερκλάσεις car, motorbike, bicycle και 2 τουλάχιστον υποκλάσεις για την κάθε υπερκλάση.

- Δημιουργήστε ένα σύστημα που να ορίζει τα interface “Comparable” και “Resizable” την abstract class Shape που θα υλοποιεί το πρώτο interface και τις υποκλάσεις Square και Circle που θα υλοποιούν το δεύτερο.

- Φτιάξτε το κατάλληλο class Diagram για τα παραπάνω συστήματα.

Προαιρετικές ασκήσεις

- Σκεφτείτε και ορίστε 2 διαφορετικά interface που θα υλοποιεί ένα σύστημα από κλάσεις,αφηρημένες κλάσεις και υποκλάσεις.
- Φτιάξετε το class Diagram για το παραπάνω σύστημα
- Δημιουργήστε την κατάλληλη main που θα προσομοιώνει το παραπάνω σύστημα δημιουργώντας instance τον αντικειμένων και κάνοντας τα να αλληλεπιδρούν μεταξύ τους

Χρήσιμα Links

- **The Java Tutorials - Trail: Learning the Java Language**
  - **Lesson: Object-Oriented Programming Concepts**
    - What Is an Object?
    - What Is a Class?
  - http://download.oracle.com/javase/tutorial/java/concepts/index.html
  - *"Object-Oriented Programming Concepts teaches you the core concepts behind object-oriented programming: objects, messages, classes, and inheritance. This lesson ends by showing you how these concepts translate into code. Feel free to skip this lesson if you are already familiar with object-oriented programming."*
- **The Java Tutorials - Trail: Learning the Java Language**
  - **Lesson: Classes and Objects**
    - http://download.oracle.com/javase/tutorial/java/javaOO/index.html
    - *"Classes and Objects describes how to write the classes from which objects are created, and how to create and use the objects."*

Χρήσιμα Links

- **The Java Tutorials - Trail: Learning the Java Language**
  - **Lesson: Object-Oriented Programming Concepts**
    - What Is Inheritance?
    - http://download.oracle.com/javase/tutorial/java/concepts/index.html
    - *"Object-Oriented Programming Concepts teaches you the core concepts behind object-oriented programming: objects, messages, classes, and inheritance. This lesson ends by showing you how these concepts translate into code. Feel free to skip this lesson if you are already familiar with object-oriented programming."*
- **The Java Tutorials - Trail: Learning the Java Language**
  - **Lesson: Interface and Inheritance**
    - http://download.oracle.com/javase/tutorial/java/IandI/subclasses.html
    - *"This section describes the way in which you can derive one class from another. That is, how a subclass can inherit fields and methods from a superclass. You will learn that all classes are derived from the Object class, and how to modify the methods that a subclass inherits from superclasses."*

Thank you