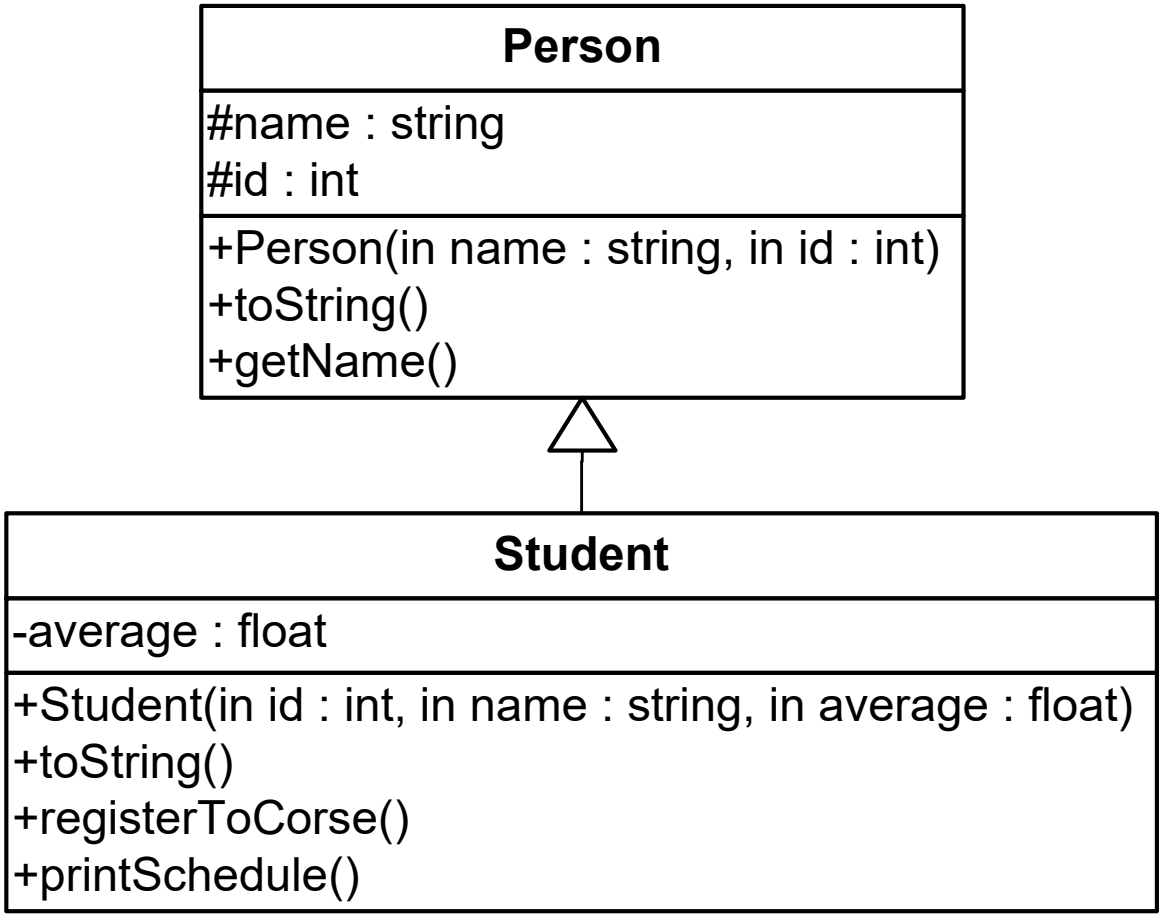
תכנות מכוון עצמים בשפת JAVA

פולימורפיזם

ביחידה זו נלמד:

- מהו פולימורפיזם
- קישור דינאמי
- אוסף פולימורפי
- מחלקות אבסטרקטיות
- שיטות אבסטרקטיות

תזכורת למחלקות בהן נשתמש



תזכורת הקוד: המחלקה Person

```
public class Person {
    protected int id;
    protected String name;

    public Person(int id, String name) {
        this.id = id;
        this.name = name;

        System.out.println("The person " + name + " is created!");
    }

    public String toString() {
        return "Id=" + id + "\t Name=" + name;
    }

    public String getName() {
        return name;
    }
} // class Person
```

תזכורת הקוד: המחלקה Student

```
public class Student extends Person {
   private float average;

   public Student(int id, String name, float average) {
       super(id, name);
       this.average = average;

       System.out.println("The student " + name + " is created!");
   }

   public String toString() {
       return super.toString() + "\t Average=" + average;
   }

   public void registerToCourse(String courseName) {
       System.out.println(name + " registers to " + courseName);
   }
} // class Student
```

יצירת אובייקט יורש מהפניה לבסיס

- מסתבר שהשורה הבאה תקינה:

Person p = new **Student**(111, “momo”, 94.2);

כאשר מייצרים אובייקט ממחלקה יורשת, ניתן להגדיר את ההפניה שלו כמחלקת הבסיס

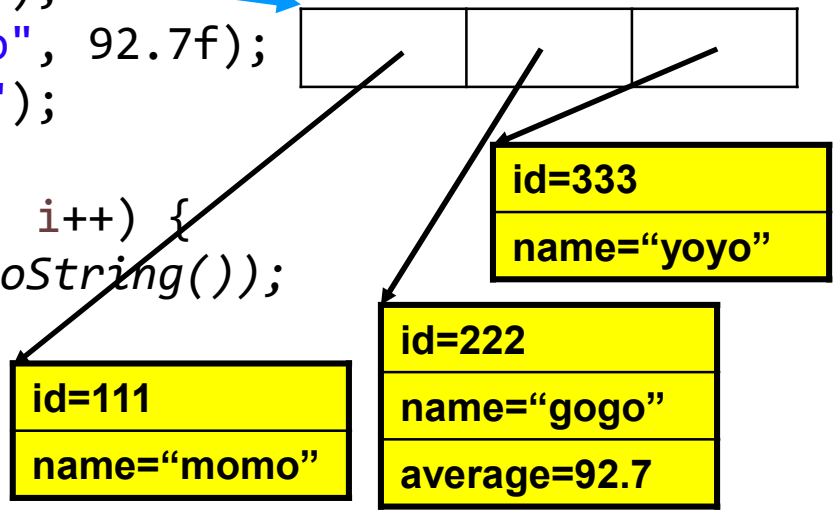
- שימושים:
  - יצירת מערך של אובייקטים מטיפוס בסיס ומטיפוס נורש ביחד
    - עד כה ראינו שבמערך כל האיברים מאותו סוג
  - שליחת אובייקט נורש לשיטה המקבלת כפרמטר אובייקט מטיפוס הבסיס

דוגמא למערך משולב

```
public static void main(String[] args) {
    Person[] persons = new Person[3];

    persons[0] = new Person(111, "momo");
    persons[1] = new Student(222, "gogo", 92.7f);
    persons[2] = new Person(333, "yoyo");

    for (int i = 0; i < persons.length; i++) {
        System.out.println(persons[i].toString());
    }
}
```




```
The person momo is created!
The person gogo is created!
The student gogo is created!
The person yoyo is created!
Id=111 Name=momo
Id=222 Name=gogo Average=92.7
Id=333 Name=yoyo
```

נשים לב שתמיד מופעלת toString
לפי טיפוס האובייקט בפועל!

דוגמה

```
public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    Person[] persons = new Person[3];

    for (int i=0 ; i < persons.length ; i++) {
        System.out.print("Enter 1 for Person, 2 for Student: ");
        int type = s.nextInt();

        switch (type) {
            case 1:
                persons[i] = new Person(111, "momo");
                break;
            case 2:
                persons[i] = new Student(222, "gogo", 93.6f);
                break;
        }
        System.out.println(persons[i].getClass().getSimpleName() + " --> "
                        + persons[i].toString());
    }
}
```

מחזיר את שם המחלקה
ממנה נוצר האובייקט

```
Enter 1 for Person, 2 for Student: 1
The person momo is created!
Person --> Id=111 Name=momo
Enter 1 for Person, 2 for Student: 2
The person gogo is created!
The student gogo was created!
Student --> Id=222 Name=gogo Average=93.6
Enter 1 for Person, 2 for Student: 1
The person momo is created!
Person --> Id=111 Name=momo
```

דוגמה זו מדגימה שהקומפיילר לא יודע
מה הטיפוס שיבחר עבור כל אובייקט
במערך, לכן הוא משאיר את הקריאה
לשימוש בשיטה לזמן ריצה

המילה השמורה instanceof

- כדי לברר אם אובייקט הוא מטיפוס מסוים, נשתמש במילה השמורה instanceof

- דוגמה:

```
if (o instanceof Person) {/*…*/}
```

- הפקודה תחזיר true אם המשתנה o הוא מטיפוס Person או יורשיו, אחרת תחזיר false

דוגמה לשימוש ב- instanceof

בפועל מועברת הפניה לאובייקט האמיתי,
אבל בשיטה ניתן לקרוא רק לשיטות מטיפוס הבסיס,
מאחר וזהו טיפוס ההפניה

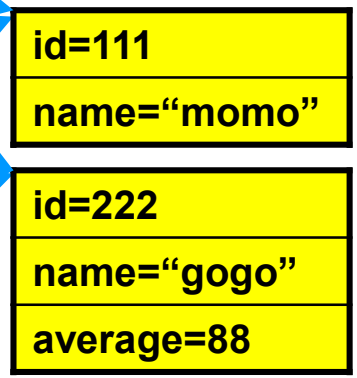
```
public static void doSomething(Person p)  {
    System.out.println("Type of p is " + p.getClass().getSimpleName());
    if (p instanceof Person)
        System.out.println(p.toString());

    if (p instanceof Student)  {
        Student s = (Student)p;
        s.registerToCourse("Java");
    }
}

public static void main(String[] args) {

    Person p1 = new Person(111, "momo");
    Person p2 = new Student(222, "gogo", 88);

    doSomething(p1);
    doSomething(p2);
}
```




```
The person momo is created!
The person gogo is created!
The student gogo was created!
Type of p is Person
Id=111 Name=momo
Type of p is Student
Id=222 Name=gogo Average=88.0
gogo registers to Java
```

מהו פולימורפיזם?

- עד כה ראינו ש:
  - אם יש הפניה לטיפוס מסוים והאובייקט בפועל הוא מטיפוס יורש:

תקרא השיטה הממומשת במחלקה של הטיפוס בפועל באופן אוטומטי

- פולימורפיזם הוא המנגנון המאפשר לקומפיילר לזהות <u>בזמן ריצה</u> מהו טיפוס האובייקט בפועל ולהפעיל את השיטה המדוייקת ביותר

קישור דינאמי

- קישור דינאמי הוא שם המנגנון שבעזרתו הקומפיילר מחליט לאיזה מימוש של השיטה לפנות בזמן ריצה, בו כבר ידוע טיפוס האובייקט בפועל
- כאשר הקומפיילר נתקל בשיטה של אובייקט מטיפוס ההפניה, הוא בודק האם עליו לחפש מימוש בעל עדיפות גבוהה יותר במחלקה שממנה נוצר האובייקט בפועל
- חידוד למתכנתי C++:
  - בשפת JAVA אין את מנגנון הלינקר, ולכן עם עליית התוכנית נטענות כל המחלקות בשימוש (Just In Time – JIT)
  - לכן הקישור הדינאמי הוא המנגנון שקורה במערכת כברירת-מחדל, ואין צורך להוסיף לתחביר דבר (אין קישור סטטי)

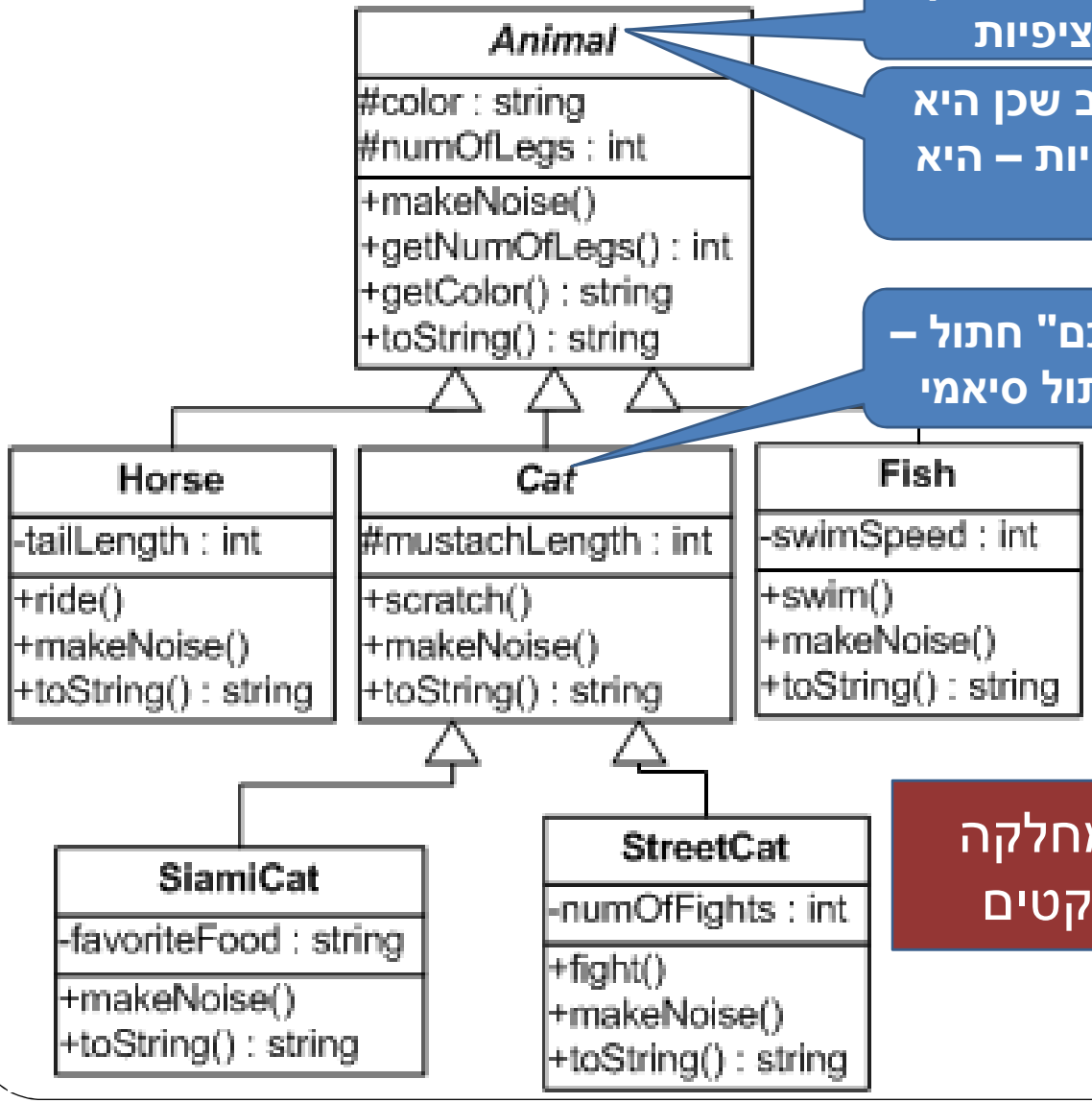
מחלקות אבסטרקטיות

בתרשים Class Diagram שם של מחלקה אבסטרקטית כתוב *בפונט מוטה (Italic)*

לא נרצה לאפשר יצירת אובייקטים מהמחלקה Animal, אלא רק של חיות ספציפיות

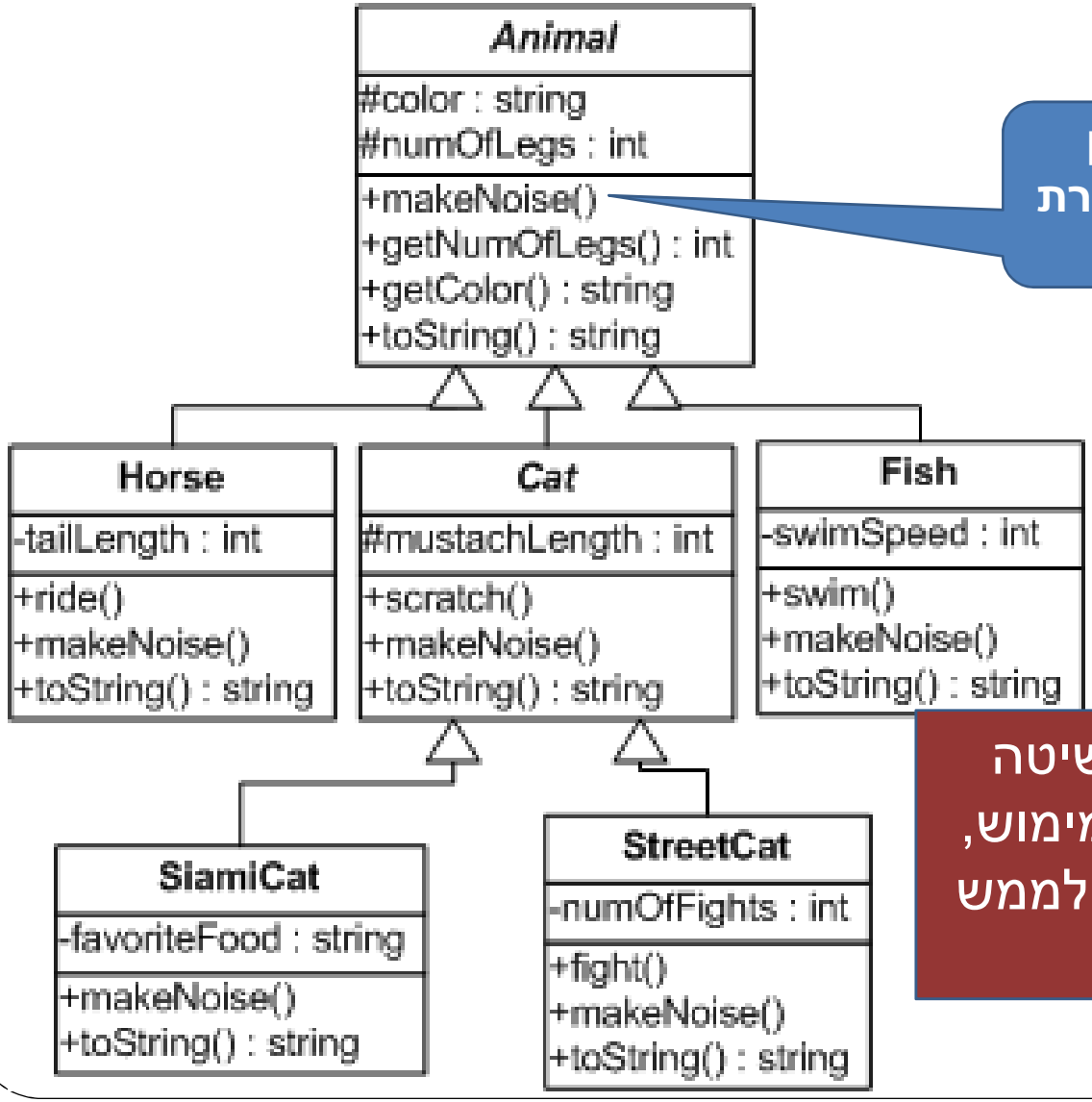
קיומה של המחלקה Animal חשוב שכן היא מכילה נתונים המשותפים לכל החיות – היא מהווה בסיס משותף

גם לא נרצה לאפשר יצירה של "סתם" חתול – כל חתול הוא או חתול רחוב או חתול סיאמי



מחלקה אבסטרקטית היא מחלקה שלא ניתן לייצר ממנה אובייקטים

שיטות אבסטרקטיות



כל חיה עושה קול ספציפי, ולכן במחלקה Animal אין מימוש ברירת מחדל למתודה makeNoise

שיטה אבסטרקטית היא שיטה שנרצה להגדיר בבסיס ללא מימוש, רק כדי להכריח את היורשים לממש אותה באופן ספציפי

דוגמאת המחלקה Animal ויורשיה: ה- Animal

```
public abstract class Animal {
    protected String color;
    protected int numOfLegs;

    public Animal(String color, int numOfLegs) {
        this.color = color;
        this.numOfLegs = numOfLegs;
    }

    public abstract void makeNoise();

    public String toString() {
        return getClass().getName() + ": color=" + color
                        + ", numOfLegs=" + numOfLegs + ", ";
    }

    public final String getColor() {
        return color;
    }

    public final int getNumOfLegs() {
        return numOfLegs;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- Horse

```
public class Horse extends Animal {
    public static final int NUM_OF_LEGS = 4;
    private int tailLen;

    public Horse(String color, int tailLen) {
        super(color, NUM_OF_LEGS);
        this.tailLen = tailLen;
    }
    @Override
    public void makeNoise() {
        System.out.println("Hi Yahah!");
    }
    public void ride() {
        System.out.println("I'm riding!");
    }
    @Override
    public String toString() {
        return super.toString() + ",  tailLen=" + tailLen;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- Cat

```
public abstract class Cat extends Animal {
    public static final int NUM_OF_LEGS = 4;
    protected int whiskersLen;

    public Cat(String color, int whiskersLen) {
        super (color, NUM_OF_LEGS);
        this.whiskersLen = whiskersLen;
    }

    public void scratch() {
        System.out.println(getClass().getSimpleName() + " is scrathing!");
    }

    @Override
    public void makeNoise() {
        System.out.println("Miyaoooooo!");
    }

    @Override
    public String toString() {
        return super.toString() + ",  whiskersLen=" + whiskersLen;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- StreetCat

```
public class StreetCat extends Cat {
    private int numOfFights;

    public StreetCat(String color, int whiskersLen, int numOfFights) {
        super(color, whiskersLen);
        this.numOfFights = numOfFights;
    }

    @Override
    public void makeNoise() {
        super.makeNoise();
        System.out.println("I want to see a dog!");
    }

    public void fight() {
        System.out.println("I want to have a good fight!");
    }

    @Override
    public String toString() {
        return super.toString() + ",  numOfFights=" + numOfFights;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- SiamiCat

```
public class SiamiCat extends Cat {
    private String favoriteFood;

    public SiamiCat(String color, int whiskersLen, String favoriteFood) {
        super(color, whiskersLen);
        this.favoriteFood = favoriteFood;
    }

    @Override
    public void makeNoise() {
        super.makeNoise();
        System.out.println("I'm so spoiled!");
    }

    @Override
    public String toString() {
        return super.toString() + ", favoriteFood: " + favoriteFood;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- Fish

```
public class Fish extends Animal {
    public static final int NUM_OF_LEGS = 0;
    private int swimSpeed;


    public Fish(String color, int swimSpeed) {
        super(color, NUM_OF_LEGS);
        this.swimSpeed = swimSpeed;
    }


    @Override
    public void makeNoise() {
        System.out.println("Blu-Blu");
    }

    @Override
    public String toString() {
        return super.toString() + ", swimSpeed: " + swimSpeed;
    }
}
```

דוגמאת המחלקה Animal ויורשיה: ה- main

```
public static void main(String[] args) {
    Animal[] animals = new Animal[4];
    animals[0] = new Horse("brown", 120);
    animals[1] = new SiamiCat("gray", 12, "RoyalCAT");
    animals[2] = new StreetCat("gingi", 15, 34);
    animals[3] = new Fish("gold", 2);

    for (int i=0 ; i < animals.length ; i++) {
        System.out.println(animals[i].toString());
        animals[i].makeNoise();
    }
}
```

אם נרצה לפנות לשיטה שקיימת רק ביורשים,
עדיין נצטרך לעשות casting...

```
Horse: color=brown, numOfLegs=4, ,  tailLen=120
Hi Yahah!
SiamiCat: color=gray, numOfLegs=4, ,  whiskersLen=12, favoriteFood: RoyalCAT
Miyaoooooo!
I'm so spoiled!
StreetCat: color=gingi, numOfLegs=4, ,  whiskersLen=15,  numOfFights=34
Miyaoooooo!
I want to see a dog!
Fish: color=gold, numOfLegs=0, , swimSpeed: 2
Blu-Blu
```

דוגמאת המחלקה Animal ויורשיה: ה- main עם פניה לפונקציה שרק ביורשים

```
public static void main(String[] args) {
    Animal[] animals = new Animal[4];
    animals[0] = new Horse("brown", 120);
    animals[1] = new SiamiCat("gray", 12, "RoyalCAT");
    animals[2] = new StreetCat("gingi", 15, 34);
    animals[3] = new Fish("gold", 2);

    for (int i=0 ; i < animals.length ; i++) {
        if (animals[i] instanceof Cat) {
            Cat temp = (Cat)animals[i];
            temp.scratch();
        }
    }
}
```

בדיקה האם האובייקט הוא Cat או אחד מיורשיו

```
SiamiCat is scrathing!
StreetCat is scrathing!
```

מחלקות ושיטות אבסטרקטיות | סיכום תחביר

- בהגדרת המחלקה נוסיף את מילת המפתח abstract

- יתכן ולא נרצה לממש את כל השיטות במחלקה זו, אלא רק להשאיר חתימה שלהן
  - לכל שיטה כזו נוסיף את מילת המפתח abstract
  - מחלקה יורשת חייבת לממש את כל השיטות האבסטרקטיות של הבסיס, אחרת גם היא תצטרך להיות מוגדרת כאבסטרקטית
  - מספיק שיש שיטה אחת אבסטרקטית במחלקה, ואז המחלקה באופן אוטומטי נחשבת לאבסטרקטית, ולכן יש לציין את המחלקה גם כ- abstract

ביחידה זו למדנו:

- מהו פולימורפיזם
- קישור דינאמי
- אוסף פולימורפי
- מחלקות אבסטרקטיות
- שיטות אבסטרקטיות

תרגיל 1: צורות

עבור כל אחת מהמחלקות הבאות יש לממש את השיטה toString המציגה את טיפוס האובייקט בנוסף לנתוניו, c'tor המאתחל את כל נתוני האובייקט, שיטות get ולספק שיטות set רק למה שהגיוני.

1. כתבו את המחלקה Shape:
   - נתוני המחלקה: עובי מסגרת וצבע
   - הגדירו את השיטות האבסטרקטיות הבאות: getArea, getPerimeter

2. כתבו את המחלקה Square היורשת מ- Shape:
   - נתוני המחלקה בנוסף הם אורך צלע הריבוע
   - כתבו את המתודה draw אשר מציגה למסך ריבוע של כוכביות בהתאם לערך שדה אורך צלע הריבוע
   - ממשו את המתודות האבסטרקטיות שהוגדרו ב- Shape (תזכורת: שטח: אורך הצלע בריבוע, היקף: אורך הצלע * 4).

3. כתבו את המחלקה Circle היורשת מ- Shape:
   - נתון המחלקה הנוסף הוא רדיוס המעגל
   - ממשו את המתודות האבסטרקטיות שהוגדרו ב- Shape (תזכורת: שטח עיגול: $\pi * r^2$ והיקף העיגול $2 * \pi * r$).

4. כתבו main:
   - שאלו את המשתמש כמה צורות ברצונו לייצר והקצו מערך בהתאם
   - עבור כל צורה שאלו את המשתמש האם זהו ריבוע או עיגול
   - עבור כל צורה הציגו נתוניה, וכן אם הצורה היא ריבוע יש גם לצייר אותו

תרגיל 2: סדרה חשבונית וסדרה הנדסית

עבור כל אחת מהמחלקות הבאות יש לממש את השיטה toString המדפיסה את טיפוס האובייקט בנוסף לנתוניו, c'tor המאתחל את כל נתוני המחלקה, שיטות get ולספק שיטות set רק למה שהגיוני.

1. כתבו את המחלקה האבסטרקטית "סדרה" שנתוניה הם האיבר הראשון בסדרה, וערך הקפיצה בין איברי הסדרה
   - הגדירו את השיטה האבסטרקטית getElement המקבלת אינדקס של איבר והשיטה תחזיר את ערכו
   - הגדירו את השיטה האבסטרקטית getSum אשר מקבלת אינדקס והשיטה תחזיר את סכום האיברים עד אותו איבר

2. כתבו את המחלקה "סדרה חשבונית" שיורשת מ"סדרה"
   - <u>תזכורת</u>: קבלת האיבר ה- n בסדרה חשבונית: כך ש- a1 הראשון, d הוא ההפרש ו- n הוא האיבר המבוקש.
   - סכום n האיברים הראשונים בסדרה חשבונית הוא

3. כתבו את המחלקה "סדרה הנדסית" שיורשת מ"סדרה"
   - <u>תזכורת</u>: קבלת האיבר ה- n בסדרה הנדסית: כך ש- a1 הראשון, q הוא המנה ו- n הוא האיבר המבוקש
   - סכום n האיברים הראשונים בסדרה הנדסית הוא

4. כתבו main הבודק את השיטות במחלקות השונות