Java lecturer 12

אז על מה נדבר היום...

- האופרטור instanceof
- מחלקות ומתודות מופשטות abstract
- Interface (אם נספיק)

RAVTECH

instanceof

המילה השמורה **instanceof** היא בעצם סוג של אופרטור שתפקידה לבדוק אם אובייקט הוא מטיפוס מסויים.
כלומר נרצה לבדוק למשל האם Student הוא גם Person.
דוגמא מוכרת:

```
@Override
public boolean equals(Object o) {
   if (this == o) return true;
   if (this == o) return true;
   if (!(o instanceof Person)) return false;
   Person person = (Person) o;
   ...
}
```

האופרטור יחזיר לנו true אם המשתנה o הוא מטיפוס Person או יורשיו, באם לא הוא יחזיר false.
אחד המקרים השימושיים. למשל בפולימורפיזם.

instanceof

בשביל הדוגמא ניצור מחלקה עבור Person ושני מחלקות יורשות.

```
public class Person {

    public void printDetails() {
        System.out.println(String.format("First name:, firstName));
    }
}

public class Teacher extends Person {

    public void showPresentation(){
        System.out.println("==== Ta Da !!! =====");
    }
}

public class Student extends Person {

    public void printAverage() {
        System.out.println(String.format("GPA: %f ", GPA));
    }
}
```

RAVTECH

instanceof

```
public static void main(String[] args) {

    Student student = new Student();
    Teacher teacher = new Teacher();

    printPersonDetails(student);
    printPersonDetails(teacher);

}

private static void printPersonDetails(Person person) {

    person.printDetails();

    if (person instanceof Student)
        ((Student) person).printAverage();

    if (person instanceof Teacher)
        ((Teacher) person).showPresentation();
}
```

במקרה הזה נרצה שהפונקציה printPersonDetails תקרא
גם למתודות שאין למחלקת האב עבור מורה ותלמיד.
בשביל זה נצטרך לבדוק על כל אובייקט מה הטיפוס שלו.
ואת זה נבצע באמצעות **instanceof**

abstract

לפעמים בירושות אנחנו נרצה ליצור מחלקת אב כלשהיא מצד אחד כדי שכמה מחלקות יוכלו לירוש ממנה אבל מצד שני כל מהות המחלקה היא רק כדי להרחיב מחלקות אחרות, ולה עצמה אין משמעות בתור מחלקה.
הכוונה היא שנרצה למנוע מהמשתמש ליצור מופע של המחלקה.
והדרך היחידה להשתמש בה היא באמצעות ירושה בלבד.
כדי לייצר מחלקה כזאת אנחנו נשתמש במילה השמורה **abstract**
לצורך הדוגמא נרצה ליצור מחלקת אב שמייצגת חיות ונקרא לה Animal.

```
public abstract class Animal {
...
}
```

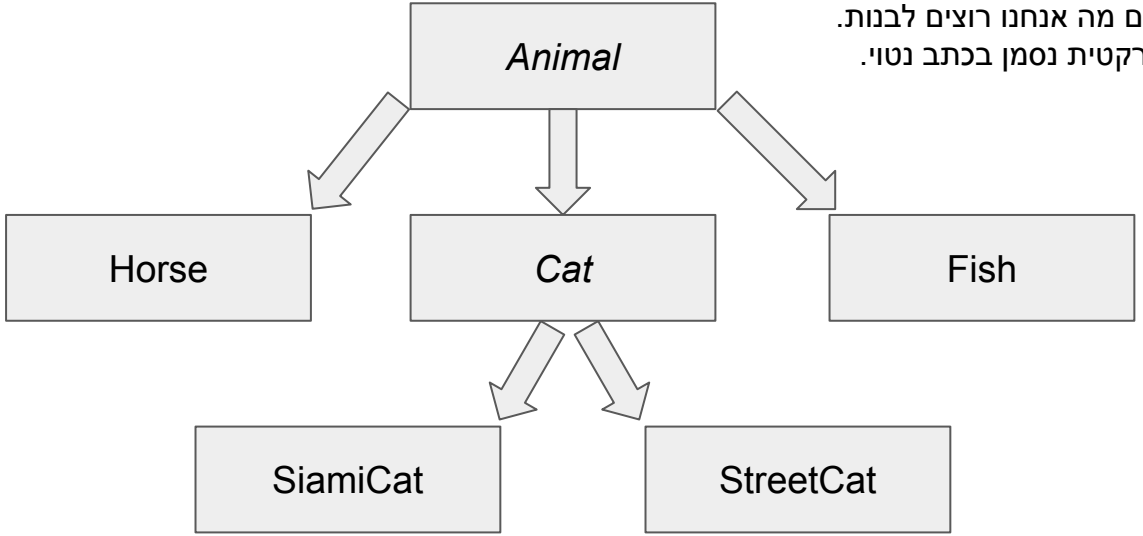
למחלקה יכולים להיות תכונות ומתודות כמו כל מחלקה רגילה, אבל המחלקה יכולה ליצור בנוסף גם מתודות אבסטרקטיות.
מתודה אבסטרקטית היא מתודה ללא מימוש, כלומר מתודה ריקה שאנחנו רוצים **להכריח** את מי שיירש את המחלקה לממש אותה באופן ספציפי.
גם כאן אופן השימוש הוא על ידי המילה **abstract.**
לדוגמא:

```
public abstract void makeNoise();
```

*במה זה שונה ממתודה רגילה?
שימו לב: מספיק שיש שיטה אחת אבסטרקטית במחלקה, ואז המחלקה באופן אוטומטי נחשבת לאבסטרקטית, ולכן יש לציין את המחלקה גם כ- **abstract**

abstract

אז איך זה נראה?
בואו נתכנן קודם מה אנחנו רוצים לבנות.
מחלקה אבסטרקטית נסמן בכתב נטוי.



עכשיו אחרי שאנחנו יודעים איזה מחלקות אנחנו רוצים, נוכל לכתוב את זה בקוד.

RAVTECH

abstract - תרגול

למחלקת Animal ניתן את התכונות הבאות:

- צבע
- מספר רגליים

שימו לב שאנחנו רוצים שהמשתנים האלה יהיו נגישים גם ליורשים.

עכשיו נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- פונקציה אבסטרקטית להשמעת קול

תרגול - abstract

פתרון:

```
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 + ", ";
   }
}
```

abstract - תרגול

למחלקת Horse ניתן את התכונה הבאה:

- אורך הזנב
- משתנה סופי ולא פרטי של מספר הרגליים

נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- פעולת רכיבה
- וכמובן הפונקציה שנהיה חייבים לממש...

תרגול - abstract

פתרון:

```
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() + ", Tail length =" + tailLen;
    }
}
```

RAVTECH

תרגול - abstract

למחלקת Fish ניתן את התכונה הבאה:

- מהירות השחייה
- משתנה סופי ולא פרטי של מספר הרגליים

נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- וכמובן הפונקציה שנהיה חייבים לממש...

תרגול - abstract

פתרון:

```
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;
    }
}
```

RAVTECH

abstract - תרגול

למחלקת Cat ניתן את התכונה הבאה:

- אורך השפם
- משתנה סופי ולא פרטי של מספר הרגליים

נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- פעולת שריטה
- וכמובן הפונקציה שנהיה חייבים לממש...

RAVTECH

תרגול - abstract

פתרון:

```
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() + ", Whiskers length =" + whiskersLen;
    }
}
```

תרגול - abstract

למחלקת StreetCat ניתן את התכונה הבאה:

- מספר הקרבות

נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- פעולת לחימה
- וכמובן הפונקציה שנהיה חייבים לממש...

תרגול - abstract

פתרון:

```
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!");
        for (int i = 0; i < 10; i++)
            super.scratch();
    }
    @Override
    public String toString() {
        return super.toString() + ",  Num of the fights=" + numOfFights;
    }
}
```

abstract - תרגול

למחלקת SiamiCat ניתן את התכונה הבאה:

- שם האוכל האהוב

נגדיר את המתודות הבאות:

- בנאי
- הדפסה (toString)
- וכמובן הפונקציה שנהיה חייבים לממש...

RAVTECH

תרגול - abstract

פתרון:

```
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;
    }
}
```

RAVTECH

תרגיל

ניצור תוכנית שתחזיק מערך של כל החיות.
התוכנית תרוץ על המערך של החיות ותבקש מכולם להשמיע קול.
עבור חתולים הם יצטרכו לשרוט בנוסף.
עבור סוס וחתול רחוב הם יצטרכו להפעיל את הפונקציות המיוחדות שלהם.

תרגיל

1. כתבו את המחלקה Shape:
   - נתוני המחלקה: עובי מסגרת וצבע
   - פעולות: בנאי, toString, get and set
   - פעולות אבסטרקטיות: getArea, getPerimeter
   -
2. כתבו את המחלקה Square היורשת מ- Shape:
   - נתוני המחלקה: אורך צלע הריבוע
   - פעולות: בנאי, toString, get and set
     מתודה draw שמדפיסה ריבוע של כוכביות בהתאם לערך שדה אורך צלע הריבוע
   - ממשו את המתודות האבסטרקטיות שהוגדרו ב- Shape

3. כתבו את המחלקה Circle היורשת מ- Shape:
   - נתון המחלקה: רדיוס המעגל
   - ממשו את המתודות האבסטרקטיות שהוגדרו ב- Shape

4. נבנה תוכנית שתבקש מהמשתמש:
   - כמה צורות ברצונו לייצר (יש לשמור את הצורות בתוך מערך)
   - עבור כל צורה שאלו את המשתמש האם זהו ריבוע או עיגול ויצרו בהתאם.
   - עבור כל צורה במערך הציגו את נתוניה, אם הצורה היא ריבוע יש גם לצייר אותו

RAVTECH