תרגול 12 – שיעורי בית

חזרה על שאלות חשובות משיעורי הבית

סטודנטים יקרים,

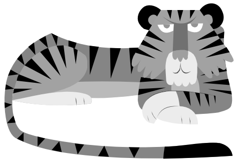
אנא הקדישו מעט מזמנכם היקר ומלאו את **סקר ההוראה**. הסקר חשוב מאד כפידבק למרצים ולמתרגלים, ומשפר את רמת ההוראה באוניברסיטה.

אנא זכרו גם שאם לא ימלאו מספיק אנשים את הסקר, לא יאכילו את הדוקטורנטים יותר.

צוות הקורס

התרגילים עליהם נעבור

Expressions (ex 8)



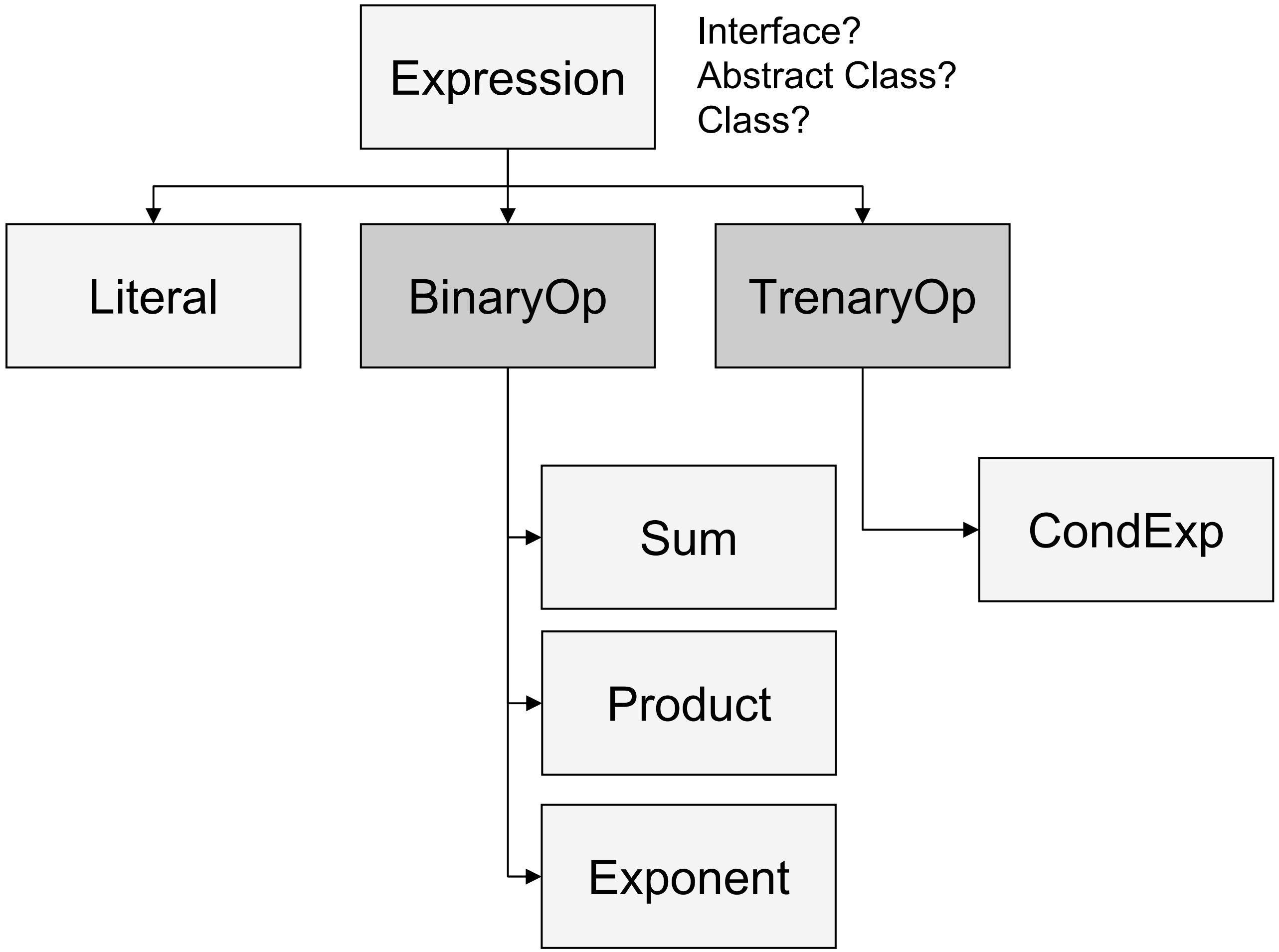
Wild World (ex 7)

Cars (ex 9)

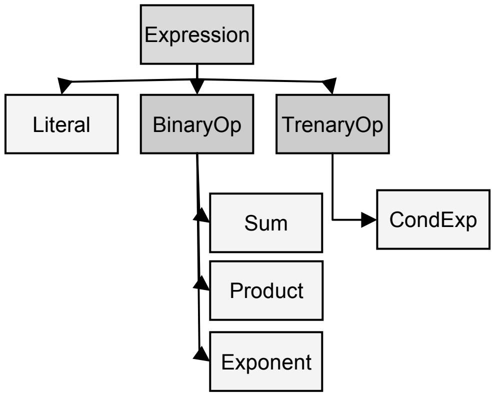
Expressions (ex 8)

- תזכורת: בתרגיל זה הייתם צריכים לממש תכנית הפותרת ביטויים אריתמטיים פשוטים
- רכיבי התכנית השונים היו
  - **Expression**– ישות המייצגת ביטוי כלשהו.
  - **Literal**– ישות המתארת מספר בודד (double).
  - **BinaryOp**– ישות המתארת פעולה בינארית (פעולה על שני ביטויים).
  - **TrenaryOp**– ישות המתארת פעולה טרינארית (פעולה על שלושה ביטויים).
  - **Sum**– ישות המתארת סכום.
  - **Product** - ישות המתארת מכפלה.
  - **Exponent** - ישות המתארת חזקה.
  - **CondExp**– ישות המתארת פעולה על שלושה פרמטרים a, b, c שהיא a ? b : c . שערוך הביטוי מתבצע כך: אם ערכו של a שונה מ 0.0 יוחזר ערכו של b אחרת יוחזר ערכו של c.

איך נראה עץ המחלקות?



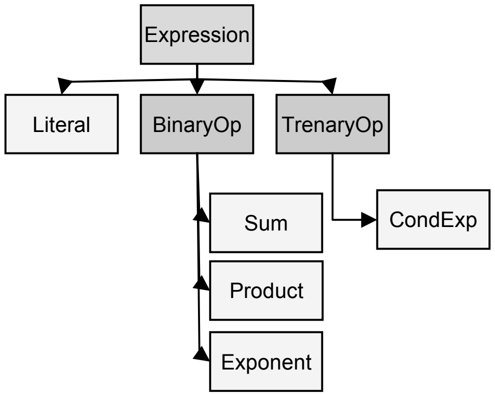
ולמימוש...




```
public interface Expression {
     public double eval();
}
```

```
public abstract class BinaryOp implements Expression {
        public BinaryOp(Expression e1, Expression e2) {
                lExp = e1; rExp = e2;
        }
        public double eval() {
                return op(lExp.eval(), rExp.eval());
        }
        abstract public double op(double left, double right);
        abstract public String op();
        public String toString() {
                return new String("(" + lExp + ") " + op() + " (" + rExp + ")");
        }
        private Expression lExp , rExp;
}
```

ולמימוש...



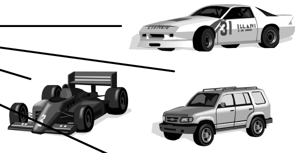

```
public interface Expression {
     public double eval();
}
```

```
public abstract class BinaryOp implements Expression {
        …
}
```

```
public class Sum extends BinaryOp {
        public Sum(Expression e1, Expression e2) { super(e1, e2); }
        public double op(double left, double right) { return left + right; }
        public String op() {
                return "+";
        }
}
```

Uma The Mechanic (ex 9)

```
public class MyCar {
     public static final int LAMBORGHINI= 2;
     public static final int SUSITA  = 4;
     public static final int RENAULT = 5;

     private int type;

     public MyCar(int type) {
          this.type = type;
     }

     public String getManufacturerName(){
          switch (type) {
          case LAMBORGHINI:
               return "LAMBORGHINI";
          case RENAULT:
               return "RENAULT";
          case SUSITA:
               return "SUSITA";
          }
          return "Manufacturer Unknown";
     }

     public int getNumOfDoors(){
          switch (type) {
          case LAMBORGHINI:
               return LAMBORGHINI;
          case RENAULT:
               return RENAULT;
          case SUSITA:
               return SUSITA;
          }
          return -1;
     }
}
```



?47

מה הבעיות במחלקה?

פתרון ראשון: ירושה

- כיצד נפתור הבעיות בעזרת ירושה?

1. מזעור כמות קוד
2. מזעור זיכרון הנדרש ע"י מופע

פתרון ראשון: ירושה

- כיצד נפתור הבעיות בעזרת ירושה?
  1. **מזעור כמות קוד**
  2. מזעור זיכרון הנדרש ע"י מופע

```
public abstract class MyCar {
     private String manufacturer;
     private int numOfDoors;
     public MyCar(String manufacturer, int numOfDoors) {
          this.manufacturer = manufacturer;
          this.numOfDoors = numOfDoors;
     }
     public String getManufacturerName(){
          return manufacturer;
     }
     public int getNumOfDoors() {
          return numOfDoors;
     }
}
```

```
public class Lamborghini extends MyCar {
     public Lamborghini() {
          super("Lamborghini", 2);
     }
}
```

פתרון ראשון: ירושה

- כיצד נפתור הבעיות בעזרת ירושה?

1. מזעור כמות קוד
2. **מזעור זיכרון הנדרש ע"י מופע**

```
public abstract class MyCar2 {
     public abstract String getManufacturerName();
     public abstract int getNumOfDoors();
}


public class Lamborghini2 extends MyCar2 {
     public String getManufacturerName() {
          return "Lamborghini";
     }

     public int getNumOfDoors() {
          return 2;
     }
}
```

פתרון ראשון: ירושה

- כיצד נפתור הבעיות בעזרת ירושה?
  1. מזעור כמות קוד
  2. מזעור זיכרון הנדרש ע"י מופע
  3. **נשתמש בטיפוס?**

```
public abstract class MyCar3 {
     public static final int LAMBORGHINI= 2;
     public static final int SUSITA  = 4;
     public static final int RENAULT = 5;

     public String getManufacturerName(){
          if (this instanceof Lamborghini3) {
               return "Lamborghini";
          } else if (this instanceof Susita3) {
               return "Susita";
          } else if (this instanceof Renault3) {
               return "Renault";
          }
          return "Manufacturer Unknown";
     }
     …
}
```

```
public class Renault3 extends MyCar3 {
}
```

```
public class Lamborghini3 extends MyCar3 {
}
```

Use of *instanceof* is bad practice!

פתרון שני: בלי ירושה?

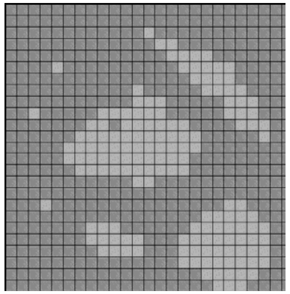
- כיצד נבטיח type safety בלי שימוש ב-enum?

```
public class MyCar4 {
     public static final int LAMBORGHINI= 2;
     public static final int SUSITA  = 4;
     public static final int RENAULT = 5;

     private int type;

     private MyCar4(int type) {
          this.type = type;
     }
     …

     public static MyCar4 createLamborghini() {
          return new MyCar4(LAMBORGHINI);
     }
     …

}
```

ולבסוף... enum

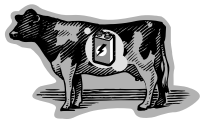
```
public enum MyCar5 {
     LAMBORGHINI(2), RENAULT(5), SUSITA(4);


     private int doors;

     MyCar5(int doors) {
          this.doors = doors;
     }

     public String getManufacturerName() {
          return this.toString();
     }

     public int getNumOfDoors() {
          return doors;
     }
}
```

```
public static void main(String[] args) {
      MyCar5 bimba = MyCar5.LAMBORGHINI;

      System.out.println(
            "bimba was manufactured by " +
            bimba.getManufacturerName());

      System.out.println(
            "bimba has " + bimba.getNumOfDoors() +
            " door" +
            (bimba.getNumOfDoors() == 1 ? "" : "s"));

      }

}
```



Wild Wild Wild World

- תזכורת
  - World הינה מחלקה המייצגת את העולם, n על n משבצות
  - בכל מהלך של הסימולציה, חיה יכולה לפעול פעם אחת
  - חיות פועלות על אנרגיה, לכל פעולה מחיר וחיה שגומרת את האנרגיה שלה, מתה



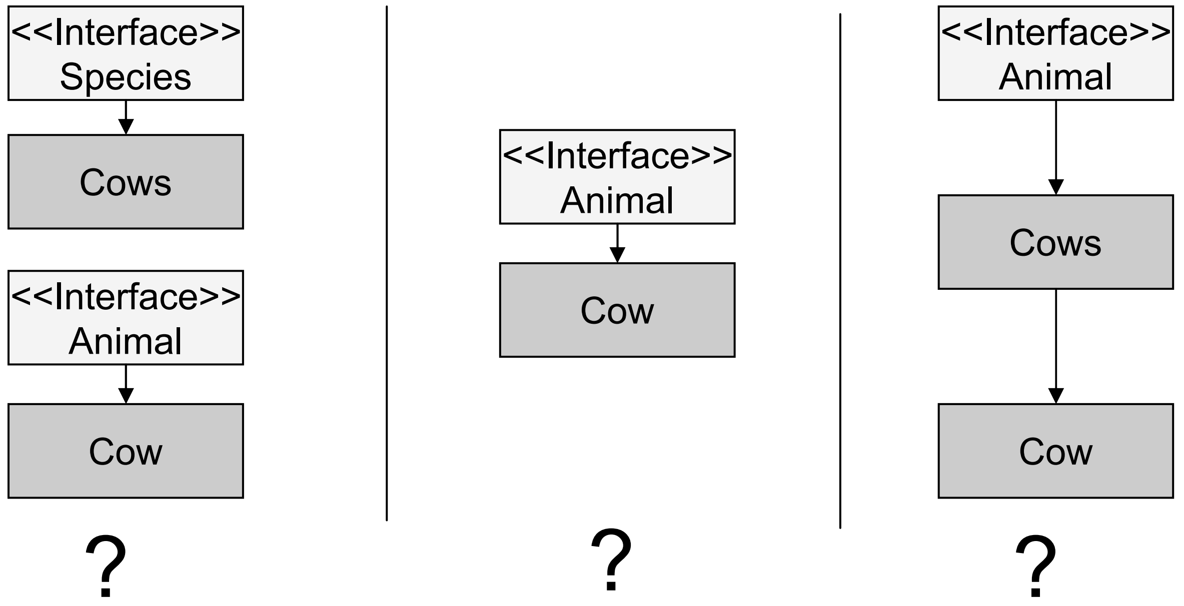
  - כל חיה שייכת למין מסוים (Species)
  - יש שני סוגי חיות: אוכלי עשב וטורפים
  - פעולות אשר חיה יכולה לעשות: לזוז, להשריץ(!)

עץ המחלקות והמנשקים

<<Interface>> Species	• מייצג מין של חיה
<<Interface>> Animal	• מייצג חיה אשר קיימת בעולם המשחק
<<Class>> World	• המחלקה הראשית, מחזיקה את "עולם המשחק"
<<Class>> Action	• מייצג פעולה אשר חיה בוחרת לעשות
<<Class>> PatchState	• מצב משבצת על לוח המשחק

חיה ומטה-חיה

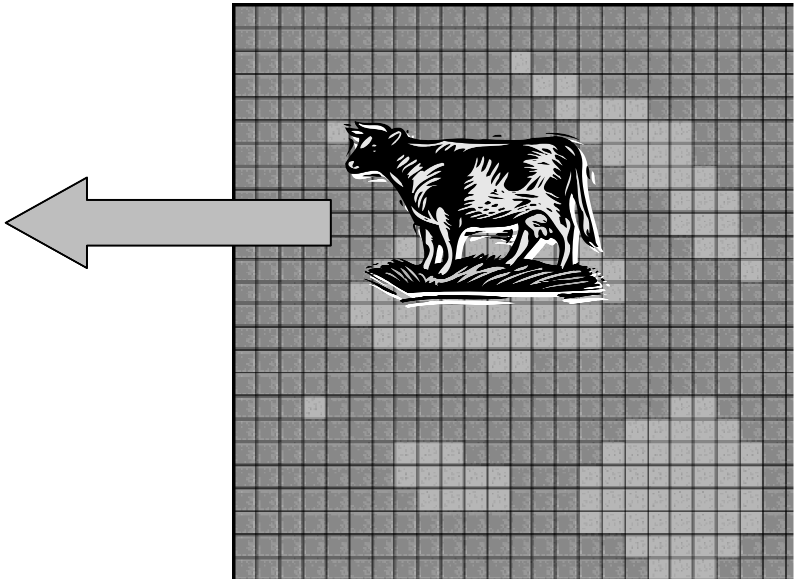
- מדוע אנו צריכים שתי מחלקות – Species & Animal ?
- האם יש דרכים נוספות לממש את היחס בין גזע וחיה?



Data Encapsulation

- אחת הנקודות החשובות בתרגיל הייתה כיצד לקבוע איזו אינפורמציה שייכת לאיזו מחלקה

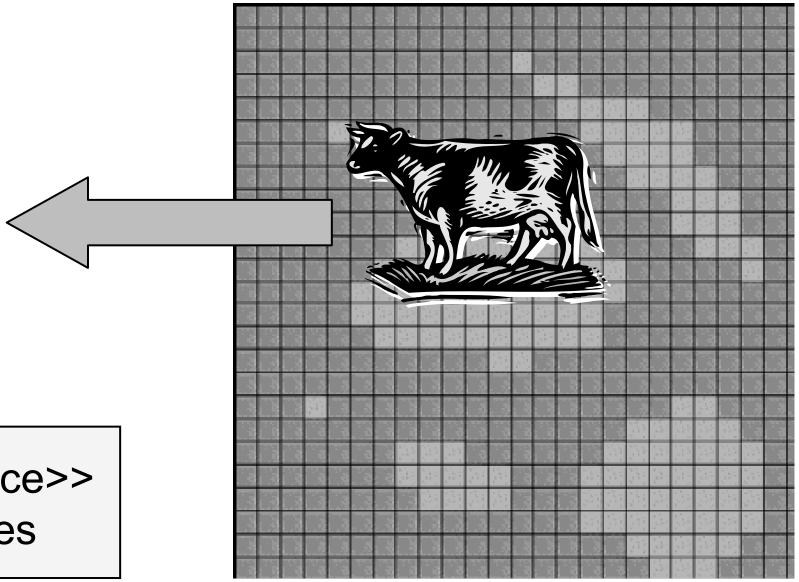
- החיה יושבת במשבצת [20,20]
- לחיה יש 5.5 יחידות אנרגיה
- החיה מסוגלת לראות 2 משבצות לכל כיוון
- העלות לבצע פעולה X היא Y

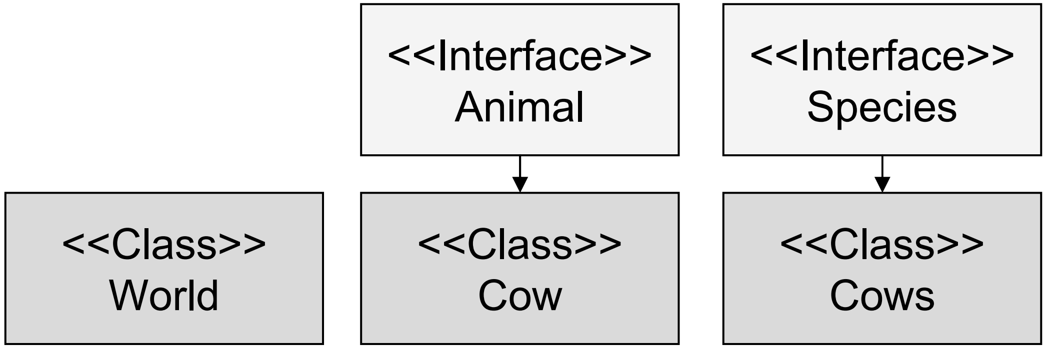


Data Encapsulation

- אחת הנקודות החשובות בתרגיל הייתה כיצד לקבוע איזו אינפורמציה שייכת לאיזו מחלקה

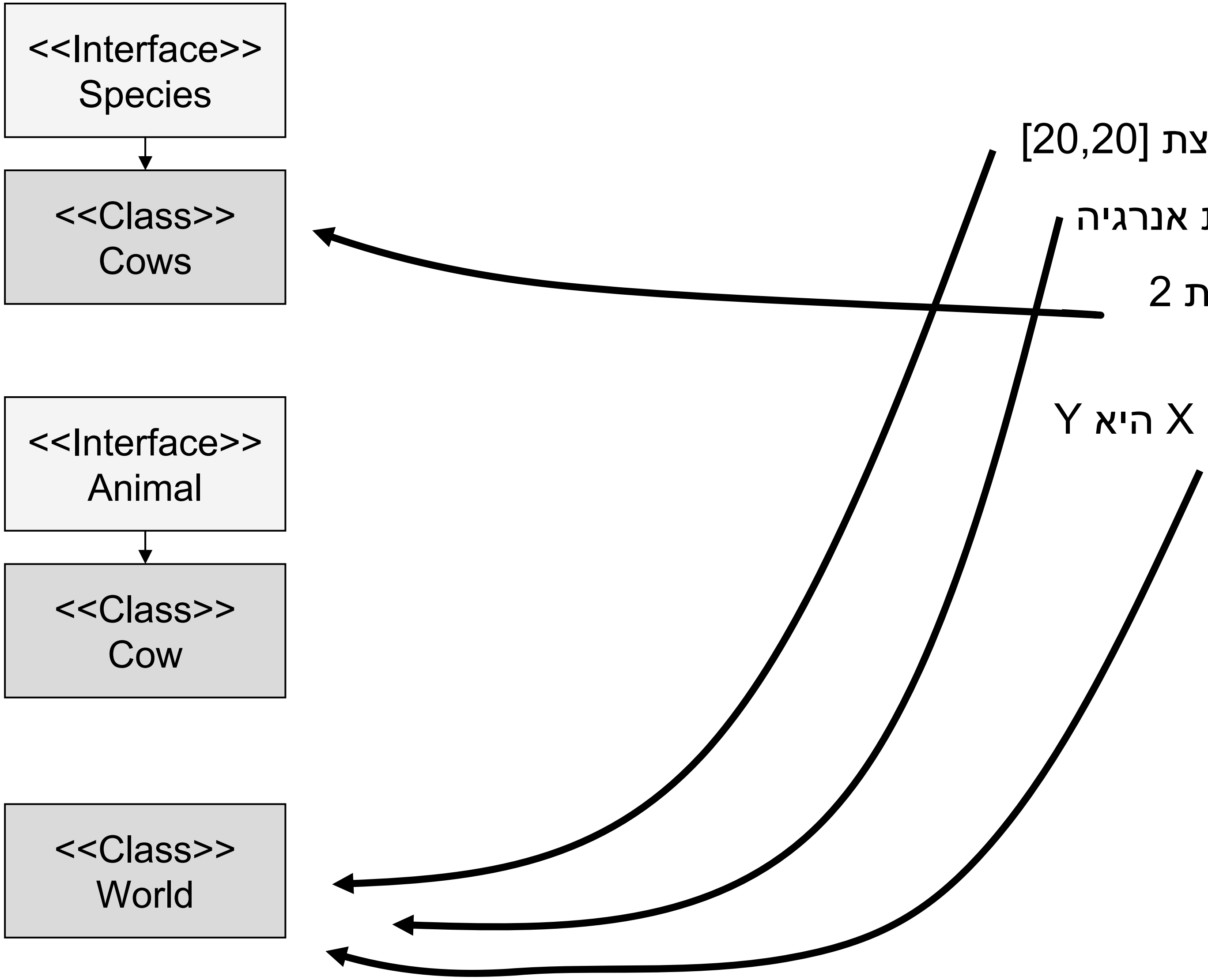
- החיה יושבת במשבצת [20,20]
- לחיה יש 5.5 יחידות אנרגיה
- החיה מסוגלת לראות 2 משבצות לכל כיוון
- העלות לבצע פעולה X היא Y





Data Encapsulation

- החיה יושבת במשבצת [20,20]
- לחיה יש 5.5 יחידות אנרגיה
- החיה מסוגלת לראות 2 משבצות לכל כיוון
- העלות לבצע פעולה X היא Y



מהלך סימולציה

- Go over list of animals (sequentially)
  1. Call Animal.Act(energy, fov)
  2. Move animal to selected square
  3. If herbivore, eat grass if exists on square
  4. If predator, check if eats a herbivore
  5. If animal spawned, insert spawn in order
- Regrow grass

אז מה חיה בעצם יודעת לעשות?

```
public interface Animal {
        public Species species();
         public Action  act(double energyLeft, PatchState[][] view);
 }
```

- לפעול, וזה כל מה שהיא צריכה לדעת
- למה?
  - מחיר כל פעולה קבוע, לא בידי החיה
  - מיקום החיה בעולם לא משנה לה, פעולתה תלויה רק בשדה ראיה הנוכחי
  - לתת לחיה לנהל את האנרגיה שלה בעצמה, סכנה ל"רמאות"

"העולם"

- אילו מבני מידע נחוצים לצורך הסימולציה במחלקה World?
- כיצד נעדכן את רשימת החיות תוך כדי מהלך אם אחת החיות השריצה, או נהרגה?
- איזה מידע נחוץ לנו על כל משבצת בעולם? האם PatchState מספיק?