

נושאים מתקדמים בתכנות מונחה עצמים ושפת JAVA

ד"ר רינה צbial-גירשין

יחידה 1

סקירה כללית של הקורס



- מעבר משפט C ו C# לשפת JAVA
- C לא OOP
- JAVA היא OOL pure וגם מאד דומה לC#
- נושאים מתקדמים בתכנות מונחה עצמים

ההיסטוריה של JAVA

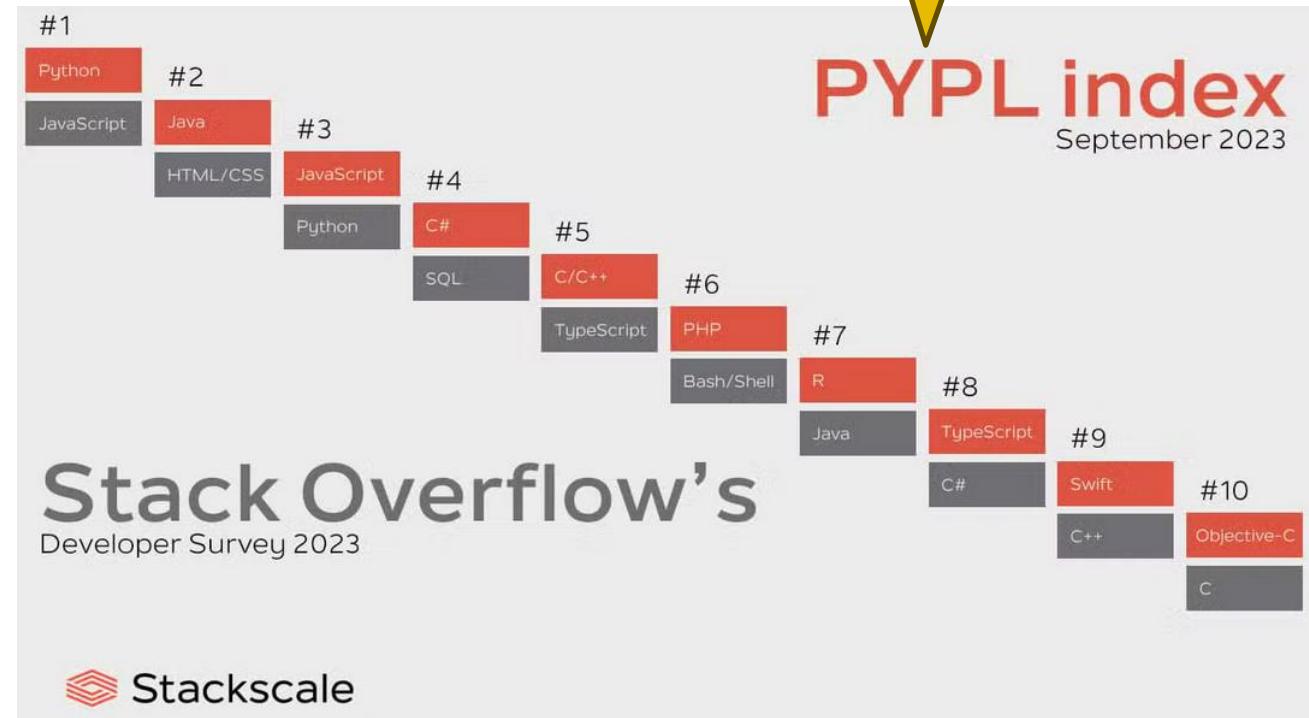
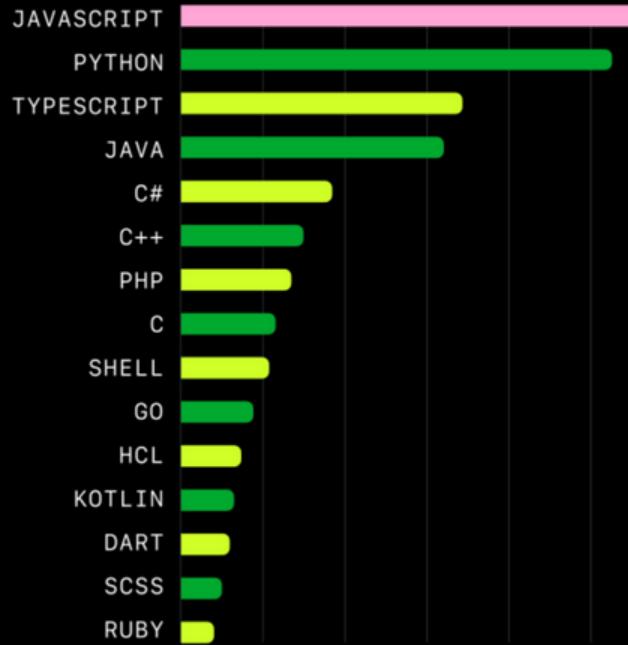


- פותחה במקור על ידי צוות קטן במעבדות Sun Microsystems בהובלת James Gosling
- 1995 שינוי שם מ Oak ל Java
- 1996 גרסה ראשונה Java1.0
- עקרון Write Once, Run Anywhere
- 2009 נרכשה על ידי Oracle

מה קורה בתחום

How often language tutorials
are searched on Google.

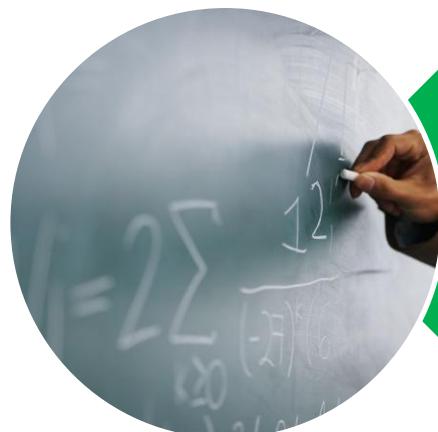
Top 10
Programming
Languages on
GitHub in 2023



פרטים טכניים



אתר הקורס – מודל



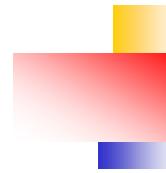
הרצאה 2 ש"ש
תרגול 2 ש"ש

פרטי מידע

- חומר נמצא באתר הקורס
<http://ruppinet.ruppin.ac.il> ■
- <https://moodle.ruppin.ac.il> ■
- ספר ■
- H.M. Deitel & P.J. Deitel, *Java: How to Program*, 11th ed. (Prentice Hall, 2018) or any previous editions
 - כל ספר אחר בעברית או אנגלית
- אתרים ■
 - אינסוף... רק לרשום "Java tutorial"



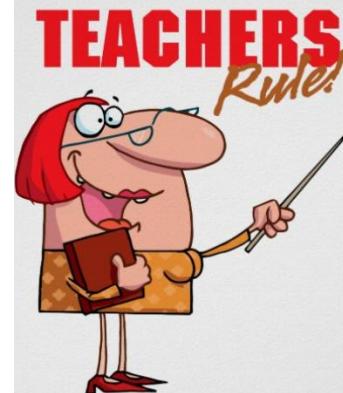
הוראה וצורות הקורס



מרצה

ד"ר רינה צbial-גירשין

rinazg@ruppin.ac.il



מתרגליים

גב מעין זנו

maayan24nasimi@gmail.com

שי אברהם

shaykos@outlook.com



Grading policy



תרגול בית

10%

חלקן חובה
עבודה בזוגות

2 בחנים

בכיתה

חובה נוכחות פיזית

20%

מבחון

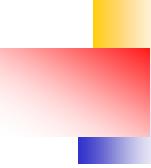
70%

מדיניות הציוניים



- בחינת סוף הסמסטר - 70%
- חובת מעבר ב מבחון
- תרגילי בית
- כל שבוע, שבועיים (5-6)
- 1-N הכי טובים
- הגשה בזוגות
- הגשה דרך האתר MOODLE וגם חובה להשתמש ב GITHUB

מפת הקורס



שבוע 3 ייחידה היברידית

1

2

3

4

5

6

מעבר בין
JAVA ל C#

רשתות
תקשורת
שרות ללקוח

תהליכי
ריבוי תהליכי
ניהול תהליכי

GUI
רכיבים
אירועים

design
patterns
tabnionot

תכנות גנרי
ואספים
עבודה עם
בסיס נתונים
JDBC

מבחן על היחידה



- Java ו-C#
- מחלקה
- סטטי
- אובייקט מוכל
- חריגות
- הורשה
- class Object
- מחלקה אבסטרקטית
- ממשק
- חריגות
- javadoc

C# מול JAVA



- קל ללמידה כל אחת מן השפות למי שיש רקע בשפה השנייה
- שתי שפות מאוד דומות
- בתחריר **syntax** (יש יותר דמיון משוני)
- מרבית מבני בקרה דומים
- שתי שפות OOP שתומכות בהורשה inheritance, encapsulation, polymorphism
- ניתן להשתמש/להרחיב הממשקים interface
- ניהול זכרון על ידי garbage collection

JAVA מול C#

2000

1995

תכונה	C# (.NET)	JAVA
OS	Windows .NET core cross-platform	Multiple
Runtime	CLR	JVM
Web Server Scripting	ASP.NET WebForms	JSF
Data Access	ADO.NET	JDBC
ORM (object-relational mapping)	Entity Framework	Hibernate (add-on)
Applications	Windows app development, game development(unity), web development	Android app development, enterprise applications, web development

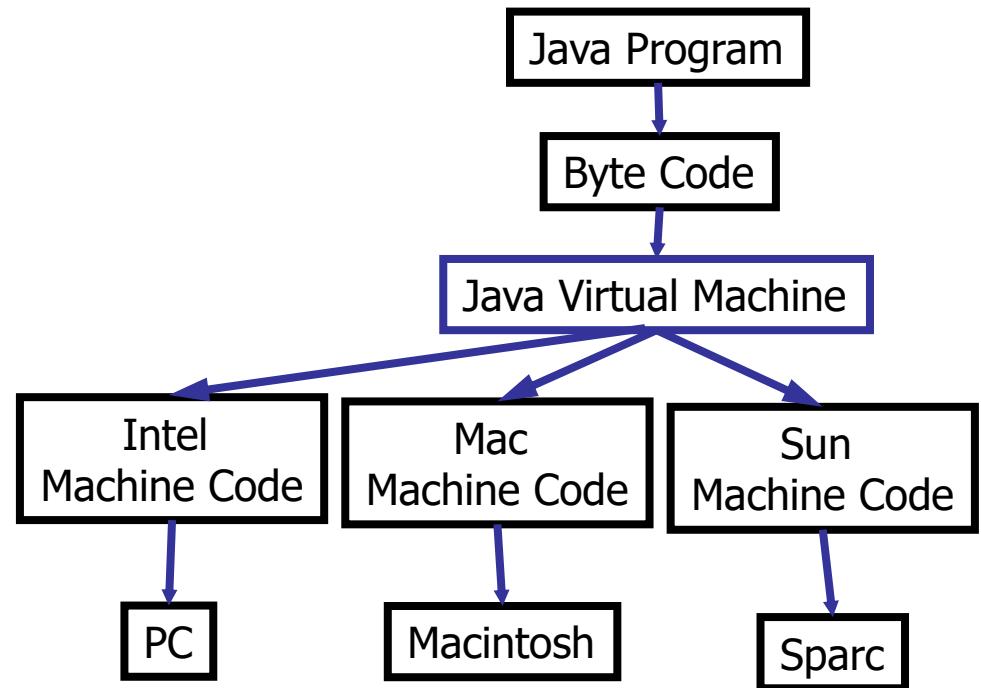
השוני



- platform independent ■ ריבוי מערכות
write once, run anywhere ■
- Libraries and Frameworks ■
- C# - .NET framework ■
- Java - set of extensive libraries and frameworks, such as Spring, ■
Hibernate, and JavaFX
- אין תכונות (getter setter) ■

מספר מילים על JVM ו Byte Code

- קוד מתורגם לבית קוד Byte Code
- למעשה זו שפת מכונה עבור JVM מכונה מדומה
- JVM מפרש interpret בית קוד לשפת מעבד עבור כל מערכת הפעלה OS



Java Main Environments



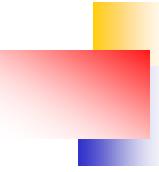
- צורת כתיבת קוד :
 - Operating System (Applications)
 - Web browser (Applets)

סביבה עבודה



- קיימות מספר סביבות עבודה
 - נשתמש ב**Eclipse**
- תהליך דו-שלבי
 - התקנת JAVA (JDK)
 - יכול להיות מותקן
 - $JDK = JRE + \text{Developer tools}$
 - $JRE = JVM + \text{library classes}$
- התקנת **Eclipse**

טיפוסים



C# (type/alias)	Java (type/wrapper)
bool / System.Boolean	boolean / java.lang.Boolean
byte / System.Byte	byte / java.lang.Byte
short / System.Int16	short / java.lang.Short
int / System.Int32	int / java.lang.Integer
long / System.Int64	long / java.lang.Long
float / System.Single	float / java.lang.Float
double / System.Double	double / java.lang.Double
char / System.Char	character / java.lang.Character

- **טיפוסים פרימיטיבים** primitive data types
 - byte, short, int, long, float, double,
 - boolean and char
- **לא פרימיטיבים** non-primitive data type or object data type
 - Classes - String, Array

פקודות קלט פלט

- עבר פקודות קלט/פלט יש להשתמש ב **System**
- **פלט**
- **קלט**
- **System.out.print()** or **System.out.println (parameters)**
- **Scanner** נמצא ב java.util |

```
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();
int age = scanner.nextInt();
```

- רצוי לסגור בסוף
- **import** ספרייה/חבילה

התוכנית הבסיסית

JAVA

```
import java.io.*;  
  
class HelloWorld  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Hello JAVA!");  
    }  
}
```

Hello JAVA!

C#

```
using System;  
  
class HelloWorld  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine("Hello C#!");  
    }  
}
```

Filename: HelloWorld.java

קבועים



private readonly int GRADE=100;

private final int GRADE=100;

■ בreadonly #C

■ ב Java final

מבנה בקרה

- `תנאי if/else`
- `ריב בחירה switch(){case X:...break;... default}`
- `לולאות while, do .. while, for`
- `אין foreach`
- `אבל for (Type element : collection) {///...}`

```
import java.util.Scanner;
```

```
public class Example
```

```
{  
    public static void main(String[] args)  
    {  
        Scanner scanner = new Scanner(System.in);  
        System.out.print("Enter the number of elements: ");  
        int size = scanner.nextInt();  
  
        int[] numbers = new int[size]; //array  
    }
```

```
for (int i = 0; i < size; i++)  
{  
    System.out.print("Enter element " + (i + 1) + ": ");  
    numbers[i] = scanner.nextInt();  
}
```

```
System.out.println("You entered the following numbers:");
```

```
for (int number : numbers) //instead of for each  
    System.out.println("Number: " + number);  
scanner.close();  
}
```

לוגמה

```
Enter the number of elements: 4  
Enter element 1: 12  
Enter element 2: 45  
Enter element 3: 56  
Enter element 4: -2  
You entered the following numbers:  
Number: 12  
Number: 45  
Number: 56  
Number: -2
```

מחרוזות



- מחרוזת **String** היא אובייקט שמייצג רצף של תווים.
- טיפוס מסוג הפניה.
- **דוגמה:**

```
String name = "Rina";
```

- בדומה לC# מחרוזת היא – **immutable** – כלומר, **בלתי ניתנת לשינוי** לאחר שהיא נוצרה.
- **דוגמה:**

```
String name = "Rina";
```

```
String upper = name.toUpperCase(); //RINA
```

דוגמה

■ דוגמה:

```
String name = "Rina";  
name = name + " Zviel";
```

- בדוגמה זו **לא שונתה** המחרוזת המקורי Rina
- **נוצרה** מחרוזת חדשה Rina Zviel ומשתנה name מצביע עליה
- המחרוזת המקורי Rina נשארה בזיכרון עד שאוסף זבל אוטומטי Garbage Collector ייאסף אותה.

מתודות נפרצות String

substring(int beginIndex, int endIndex) –
מחזירת מחרוזת חיתוך חלק

String word = "Programming";
System.out.println(word.substring(0, 6)); // Progra

equals() – השוואת מחרוזות לפי תוכן
String a = "Java";
String b = "java";
System.out.println(a.equals(b)); // false
System.out.println(a.equalsIgnoreCase(b)); // true

charAt(int index) –
קבלת תו לפי מיקום
String s = "Code";
System.out.println(s.charAt(1)); // o

- length() –
אורך המחרוזת (מספר תווים)
String name = "Rina";
System.out.println(name.length()); // 4
- trim() –
הסרת רווחים מיוטרים בתחילת ובfine של המחרוזת
String text = " hello world ";
String trimmed = text.trim();
System.out.println(trimmed); // "hello world"
- toUpperCase() –
המרת כל האותיות לאותיות גדולות
String msg = "good hour";
System.out.println(msg.toUpperCase()); // GOOD HOUR
- toLowerCase() –
המרת כל האותיות לאותיות קטנות
String msg = "HELLO";
System.out.println(msg.toLowerCase()); // hello

מחלקה `StringBuilder`

- מחרוזת היא בלתי ניתנת לשינוי – כאשר `StringBuilder` היא מחלוקת `immutable`ニアור – אשר ניתנת לשינוי.
- באמצעות מחלוקת `StringBuilder` ניתן לשנות את התוכן של המחרוזת בלי ליצור אובייקט חדש בכל פעם.
- **דוגמה:**

```
StringBuilder sb = new StringBuilder();
sb.append("Hello");      // מוסיף טקסט לסוף
sb.append(" ");
sb.append("World!"); // Hello World!
```

יעילות

```
public class StringTest
{  public static void main(String[] args)
{
long start = System.currentTimeMillis();
String result = "";
for (int i = 0; i < 10000; i++)
    result += i; // new string

long end = System.currentTimeMillis();
System.out.println("Time using String: " + (end -
start) + " ms");
}
}
```

```
public class StringTest
{  public static void main(String[] args)
{
long start = System.currentTimeMillis();
StringBuilder sb = new StringBuilder();
for (int i = 0; i < 10000; i++)
    sb.append(i); // the same string

long end = System.currentTimeMillis();
System.out.println("Time using String: " + (end -
start) + " ms");
}
}
```

יותר מהירה כי היא לא יוצרת אובייקט חדש בכל איטרציה.

יעילות



- כאשר יש קוד שמחבר הרבה מחרוזות (כמו בתוך LOLA), השימוש ב- **String** יוצר אובייקט חדש בכל פעם, מה שפוגע ביצועים.
- לעומת זאת, **StringBuilder** משנה את התוכן באותו אובייקט, ולכך היא יעילה יותר בזיכרון ובמהירות

מתודות נפרצאות StringBuilder

תיאור	מתודה
מוסיף טקסט לסוף	append(String s)
מוסיף טקסט במקומות מסוימים	insert(int offset, String s)
מוחק תווים בין מיקומים	delete(int start, int end)
הופך את סדר התווים	reverse()
מחליפ תווים בתחום מסוימים	replace(int start, int end, String s)

יותר מהירה כי היא לא יוצרת אובייקט חדש בכל איטרציה.

מַעֲרְכִים



- אין הבדל בהכרזה, גישה

```
int[] numbers = new int[] { 1, 2, 3, 4, 5 }; //or  
int[] numbers = { 1, 2, 3, 4, 5 }; //or  
int[] numbers;  
numbers = new int[5];  
...  
numbers[2] = 10;
```

length JAVA Length #C ■

```
String[] names = {"Rina", "Avner", "Beni", "Arava"};  
for (int i = 0; i < names.length; i++)  
    System.out.println(names[i]);
```

לוגמה

```
public class mainClass
{
    public static void main(String[] args)
    {
        String[] names = {"Rina", "Avner", "Beni", "Arava"};
        for (int i = 0; i < names.length; i++)
            System.out.println(names[i]);
    } //end of main
} //end of mainClass
```

Rina
Avner
Beni
Arava

מערך רב ממדי



- **תחביר**

```
dataType[1st dim][2nd dim][][]..[Nth dim] aName = new datatype[size1][size2]....[sizeN];
```

- **דוגמה :**

```
int[][][] arr3D = new int[10][20][30];
```

- ניתן ליצור מערך רב ממדי בגודל לא סימטרי

```
int[][] oddNumbers = { {1, 7}, {9, 11, 13, 15}, { 19, 21, 23} };
```

לוגמה

```
public class mainClass
{
    int[][] numbers = { {1, 7},
                        {9, 11, 13, 15}, { 19, 21, 23} }; //different size

    for(int i = 0; i < numbers.length; i++)
    {
        for(int j = 0; j < numbers[i].length; j++)
            System.out.print(numbers[i][j] + " ");
        System.out.println( );
    }
} //end of main
} //end of mainClass
```

```
1 7
9 11 13 15
19 21 23
```

מערך דינامي ArrayList



Array



```
ArrayList<TYPE> names = new ArrayList<TYPE>();
```

```
names.add(NewItem);
```

```
names.get(0);      names.set(0, "Beni");
```

```
names.size();
```

- מערך דינامي בעל גודל משתנה ArrayList
- נמצא בCHASELIST.java.util
- יש ליצור (חשוב לציין טיפוס האיברים)

להוסיף איברים

ניתן לגשת לאיבר במקומות i get/set

יש לו גודל

לוגמה

```
public class mainClass
{
    public static void main(String[] args)
    {
        ArrayList<String> names = new ArrayList<String>();
        names.add("Anver");
        names.add("Rina");
        names.add("Shiri");
        System.out.println(names);
        System.out.println("The size is "+ names.size());
        System.out.println("Contains Beni? "+names.contains("Beni"));
        names.clear(); //clears
    } //end of main
} //end of mainClass
```

```
[Anver, Rina, Shiri]
The size is 3
Contains Beni? false
```

מחלקה פשוטה



- מחלקה שיוצרת אובייקט

```
public class NAME  
{  
    //data fields;  
    //methods;  
}
```

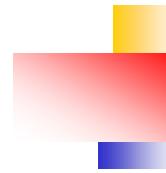
- הרשות גישה - ברירת מחדל לא פרטי "חכימתי" כל המחלקות באותו חבילה רואות את השדה

חבילת/מארז



- JAVA מגיעה עם אוסף גדול של מחלקות סטנדרטיות לשימוש המתכנת.
- המחלקות מאורגנות במבנה של מארז (package).
- החבילה/המארז **java.lang** מכילה אוסף מחלקות בסיסיות של השפה.
 - למשל את המחלקות **System, Math, String**
 - מארז זה **מודול אוטומטי** בכל תוכנית בגיאו.
- אם רוצים לכלול גם מחלקות ממאזים אחרים, יש להשתמש בפקודה **import**
- **דוגמה :**
import java.util.ArrayList;

Package vs namespace



JAVA

מארגן קבצים פיזיים

```
package SomeExample;  
  
public class Example  
{  
}
```

similar

C#

מארגן יחידות לוגיות
כגון מחלקות וממשקיים

```
namespace SomeExample;  
  
public class Example  
{  
}
```

- לא חובה להגדיר מארז
- אבל אם מגדילים אז זו צריכה להיות שורה ראשונה בקובץ
- אין קיון nesting של החבילות

מחלקה



Toy	
+nextID	: int
-name	: String
-id	: int
-price	: double
+toString()	: String
+getName()	: String
+setName(name : String)	: void
+getId()	: int
+setId(id : int)	: void
+getPrice()	: double
+setPrice(price : double)	: void
+activateFeatures()	



מחלקה פשוטה

Toy

```
public class Toy
{
    // Static variable to keep track of the next ID to be assigned
    private static int nextId = 1;

    private String name;
    private int id;
    private double price;

    public Toy(String name, double price)
    {
        this.name = name;
        setPrice(price); // Use setter to ensure validation
        // Assign the next available ID and increment it
        this.id = nextId++; //no setID
    }

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public int getId()
    {
        return id;
    }
}
```

```
public double getPrice()
{
    return price;
}

public void setPrice(double price) throws IllegalArgumentException
{
    // If negative throw Exception
    if (price < 0)
        throw new
            IllegalArgumentException("Price cannot be negative.");
    this.price = price;
}

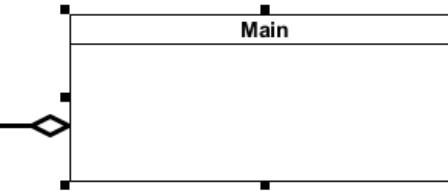
public void activateFeatures()
{
    System.out.println("Activating basic features of the toy: " + name);
}

@Override public String toString()
{
    return "Toy: " + name + " - ID: " + id + " - Price: $" + price;
}
} //end of class
```

main

```
public class mainClass
{
    public static void main(String[] args)
    {
        try {
            // Create some Toy objects
            Toy toy1 = new Toy("Dog", 25.0);
            System.out.println(toy1);
            Toy toy2 = new Toy("Dog", -45.2);
            System.out.println(toy2);
        }
        catch (IllegalArgumentException e)
        {
            System.out.println("Exception caught: " + e.getMessage());
        }
    } //end of main
} //end of mainClass
```

Toy
+nextID : int
-name : String
-id : int
-price : double
+toString() : String
+getName() : String
+setName(name : String) : void
+getId() : int
+setId(id : int) : void
+getPrice() : double
+setPrice(price : double) : void



```
Java Application - JRE Java(TM) SE Runtime Environment (Java 1.8.0_111)
Toy: Dog - ID: 1 - Price: $25.0
Exception caught: Price cannot be negative.
```

main

```
public class MainClass
{
    public static void main(String[] args)
    {
        // Create some Toy objects
        Toy toy1 = new Toy("Dog", 25.0);
        System.out.println(toy1);
        Toy toy2 = new Toy("Dog", -45.2);
        System.out.println(toy2);
    } //end of main
} //end of mainClass
```

כתב לא מסודרת
ללא try catch

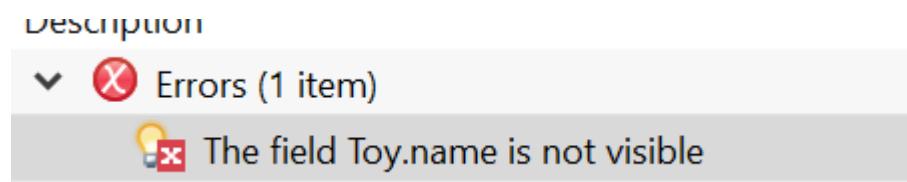
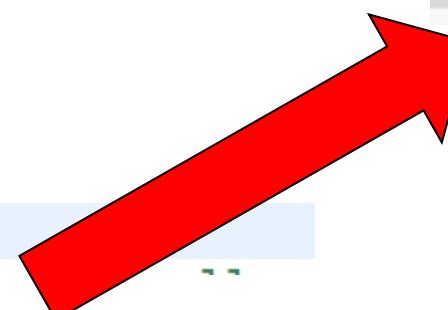
```
<terminated> app [Java Application] C:\Users\paral\Downloads\eclipse-java-2020-12-R-win32-x86_64\ec
Exception in thread "main" Toy: Dog - ID: 1 - Price: $25.0
java.lang.IllegalArgumentException: Price cannot be negative.
        at Toy.setPrice(Toy.java:82)
        at Toy.<init>(Toy.java:34)
        at app.main(app.java:14)
```

גישה לשדה ב**main**

```
public class Toy
{
    ...
    private String name;
    double price; //not private
```

להלן **main**

```
// Create some Toy objects
Toy toy1 = new Toy("Dog", 25.0);
System.out.println(toy1.price);
System.out.println(toy1.name);
```



מספר מילים על מחלקות



- מחלקות של רמה עליונה top-level הם
 - או public או כלום (non public)
- לא ניתן ליצור מחלוקת (לא פנימית inner) שהיא פרטית private
- אם לא רשומים כלום או חברי מחלוקת members ניתנים לגישה רק לחבילה/מארז
- **מכלול** : בירית מחדל של הרשות גישה - לא פרטי "חבורה"

One per class

מספר מילימ על static



- משתנה סטטי – אחד משותף לכל האובייקטים של המחלקה

```
private static int nextId = 1;  
this.id = nextId++;
```

- מетод סטטית – מトודה שמייצת לכל המחלקה ולא לאובייקט ספציפי

```
public static int MethodName(int number)  
{...}
```

- שימוש/קריאה

```
className.MethodName();
```

- מותר להשתמש בדרך אובייקט (אבל לא מומלץ) warning

- **ב# אסור** להפעיל מtodה סטטית דרך אובייקט.

דוגמָה

```
public class Utility
{
    // Static variable
    public static final double PI = 3.14159;
```

```
// Static method
public static int square(int number)
{ return number * number; }
```

```
public static void main(String[] args)
{
    // Accessing static variable
    System.out.println("Value of PI: " + Utility.PI);
```

```
// Calling static method
int result = Utility.square(5);
System.out.println("Square of 5: " + result);
}
} //end of class
```

Value of PI: 3.14159
Square of 5: 25

מחלקה סטאטית



- מחלקת סטאטית חייבת להיות פנימית מקוננת
- ניתן לגשת רק לחברים סטאטיים של מחלקת חיצונית

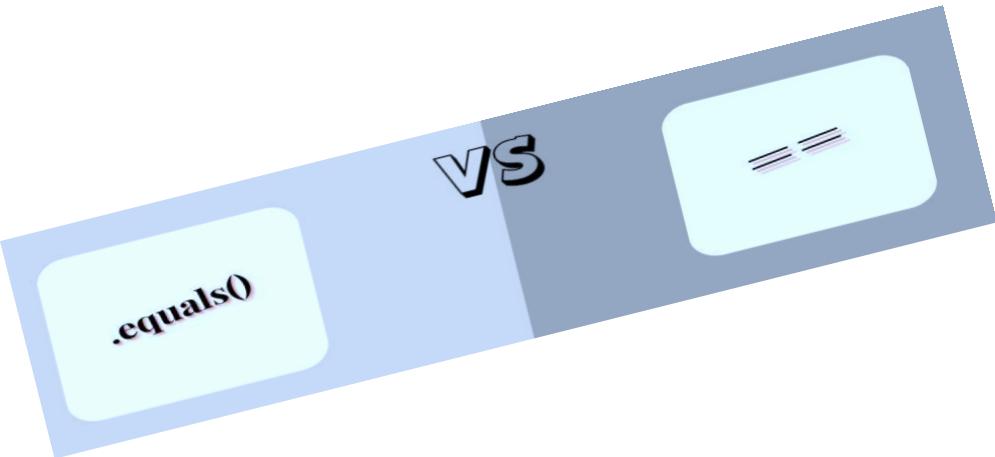
```
class OuterClass
{
    static class NestedDemo
    {
    }
}
```

```
public class StaticExample
{
    private static String s= "Static String";

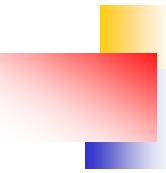
    static class NestedDemo
    {
        //non-static method of the nested class
        public void show()
        { System.out.println(s); }
    } //end of NestedDemo class
}
```

```
public static void main(String args[])
{
    StaticExample.NestedDemo obj = new StaticExample.NestedDemo();
    //invoking the method of the nested class
    obj.show();
}
} //end of class
```

```
<terminated> StaticExam
Static String
```



`==` vs `equals`



- ב C# `==` משווה וניתן להשתמש בו במקום `equals` בזאת
- ב JAVA `==` מיציר תוצאות שונות ולכון להשתמש במתודה `equals` (חוץ ממחלקה `String`)
- אופרטור `==` משווה כתובות בזיכרון
- מתודה `equals` משווה תוכן

Exceptions



- חריגה – שגיאת הריצה (הקוד התקומפל אבל ...
 - זורקים חריגה `throw`
 - תופסים חריגה `catch`
 - בлок מסכם `finally`
 - קורה תמיד
- בחתימה של מетодה שזורקת חריגה ולא תופסת צורך לכתוב `throws`
`public type method() throws SomeException {}`
`public void setPrice(double price) throws IllegalArgumentException {}`

```

import java.util.Scanner;

public class ExcExample
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter which element to print: ");
        int num = scanner.nextInt();
        try {
            int[] myNumbers = {1, 2, 3};
            System.out.println(myNumbers[num]);
        } catch (Exception e) {
            System.out.println("Something went wrong.");
        } finally
        {
            System.out.println("The 'try catch' is finished.");
        }
        scanner.close();
    }
} //end of class

```

חריגת

Enter which element to print: 2
3
The 'try catch' is finished.

Enter which element to print: 6
Something went wrong.
The 'try catch' is finished.

C#

```
public class A : B  
{  
    ...  
    base();  
    ...  
}
```

```
public class A extends B  
{  
    ...  
    super();  
    ...  
}
```

הוֹרֶשֶׁה

- אב יחיד - single inheritance
- יש להרחיב מחלקה בסיס extends
- קרייה למחלקה בסיס באמצעות super
- **תחביר:**

לוגמה להורשה בסיסית

JAVA

```
public class A
{
    public A ( string name)
    {
        {
    }
}

public class B extends A
{
    public B ()
    { this ("Rina – Java" , 0 ); }

    public B (string name, int type)
    { super(name); }
}
```

C#

```
public class A
{
    public A ( string name)
    {
        {
    }
}

public class B:A
{
    public B (): this ("Rina – C#" , 0 )
    {
    }

    public B (string name, int type): base(name)
    {
    }
}
```

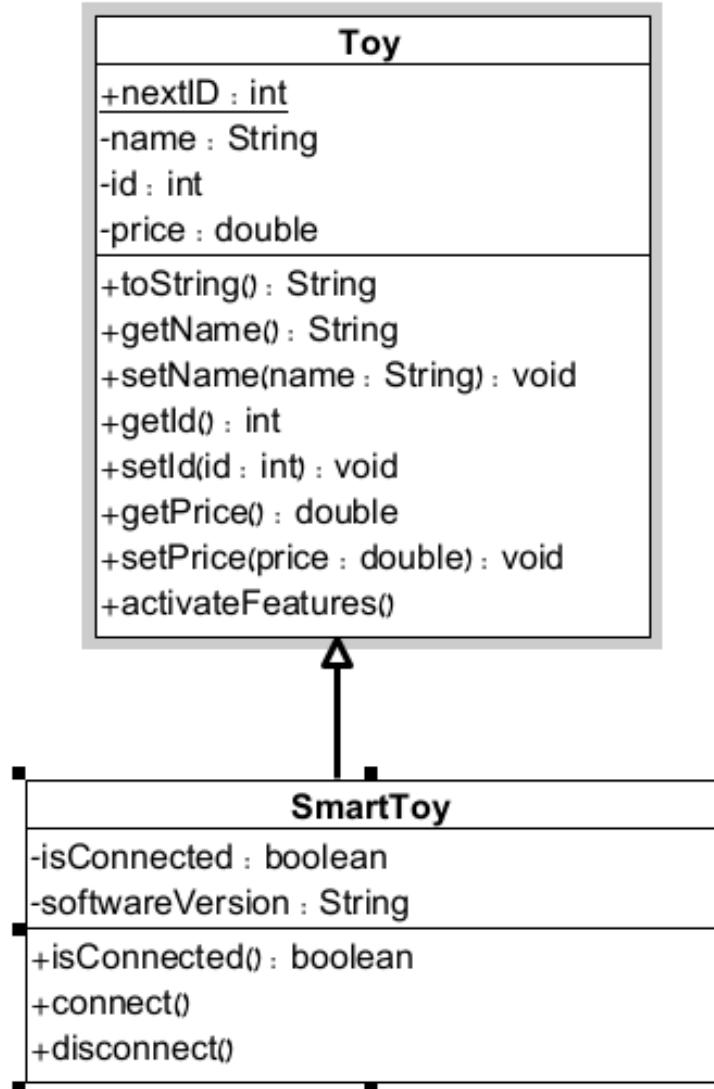
מתודה מדומה



- כל המethodות בJAVA מדומות תמיד (ברירת מחדל default).
- לכן תמיד ניתן לדרוס אותו override
- במקרה של דרישת מקובל לרשום

@Override

חזרה למחלקה Toy



חנות הצעוצים "us Toys" מכירה עד כה בעיקר צעצועים קלאסיים עם מאפיינים בסיסיים – כגון בובות, מכוניות צעצוע וצעודי בעלי חיים. כל צעצוע מקבל מידע ייחודי, שם ומחיר. עם הזמן, החנות התחליה למכור צעצועים חכמים המחברים לאינטרנט, בעלי גרסת תוכנה וכוללים תכונות אינטראקטיביות, כמו שליטה מרוחק.

לדוגמה, צעצוע מסווג **Smart Car** הוא מכונית שאפשר לחבר לאפליקציה, ו- **Smart Robot** הוא רובוט שמסוגל להתחבר לאינטרנט ולתקשך עם הילד. עבור צעצוע חכם יש לשמור האם מחובר לאינטרנט (כן/לא) ואת גרסת התוכנה המותקנת עליו.

הורשה

```
public class SmartToy extends Toy
{
    private boolean isConnected;
    private String softwareVersion;

    public SmartToy(String name, double price, String
softwareVersion)
    { super(name, price);
        this.softwareVersion = softwareVersion;
        this.isConnected = false; }

    public boolean isConnected()
    { return isConnected; }

    public void connect()
    { isConnected = true; }

    public void disconnect()
    { isConnected = false; }

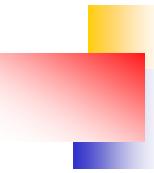
    public String getSoftwareVersion()
    { return softwareVersion; }
```

```
public void updateSoftware(String newVersion)
{ this.softwareVersion = newVersion;
    System.out.println("Software updated to version: " +
newVersion);}

@Override
public void activateFeatures()
{ if (isConnected)
    System.out.println("Activating smart features for " + getName()
+ ": Online mode enabled.");
    else System.out.println("Activating smart features for " + getName() + ": Please connect to enable online features."); }

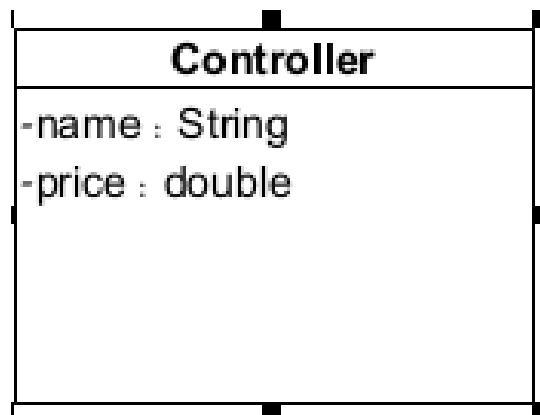
@Override public String toString()
{ return super.toString() + " - Smart Features [Connected: " +
isConnected + ", Software Version: " + softwareVersion + "]"; }
} //end of class
```

המשך הסיפור



בעקבות הביקוש הגובר לצעוצועים חכמים, הchnות הchallenge לשוק בקרי שליטה שונים שמותאימים במיוחד לצעוצועים החכמים.

בקרי השליטה, כמו Raspberry Pi ו- Arduino מאפשרים להפעיל פונקציות מתקדמות לצעוצועים החכמים ולהפוך אותם ל互動קטיביים. כל בקר מגיע עם מחיר משלו ושם הבקר.



```

public class Controller
{
    private String name; // The name of the controller
    private double price; // The price of the controller

    public Controller(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public double getPrice() { return price; }

    public void setPrice(double price) { this.price = price; }

    @Override public String toString()
    { return "Controller: " + name + " - Price: $" + price; }
}

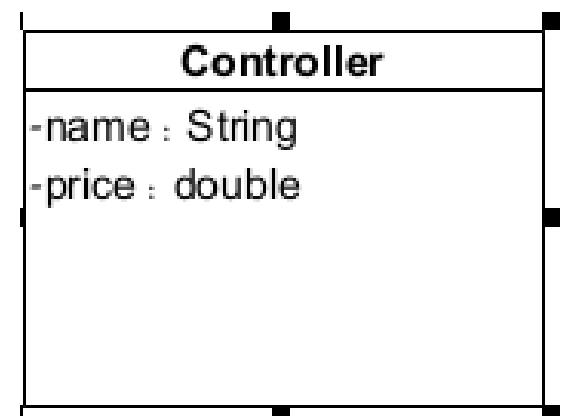
```

עוז מחלקה

```

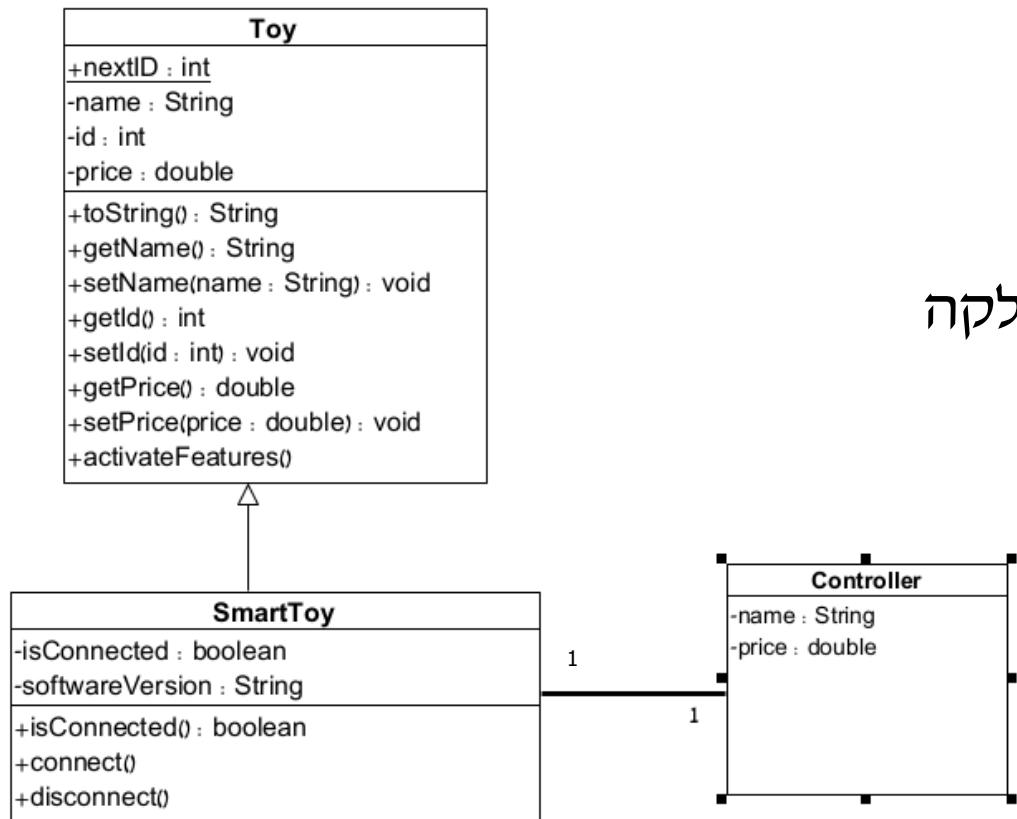
// Create some Controller objects
Controller controller1 = new Controller("Arduino", 15.4);
Controller controller2 = new Controller("RasberryPie", 134.5);

```



המשך הסיפור

לאחר שהחנות "as is" החלה לשוק בקרי שליטה חכמים, הלקוחות גילו את היכולת לשדרג את הצעועים החכמים שלהם ולהפוך אותם לאינטראקטיביים יותר. כעת יכול לשדרג את הצעוע החכם שקנה עם בקרים מתקדם כדי לאפשר פונקציות משחק ולהעצים את חווית המשחק. ברגע שמחברים את הבקר לצעוע החכם, הצעוע מתעדכן במחירו כך שיילול את מחיר הבקר, ומאפשר גישה לתוכנות של הבקר.



ישנה היכרות (קשר הכרות) בין מחלקה Controller ומחלקה Smart Toy - יבוא לידי ביטוי בקוד ע"י מחלוקת

מוכלת

```
public class SmartToy extends Toy
{
    private Controller controller;
    private boolean isConnected;
    private String softwareVersion;
```

```
public SmartToy(String name, double price, Controller controller, String
softwareVersion)
{   super(name, price);
    this.controller = controller;
    this.softwareVersion = softwareVersion;
    this.isConnected = false;
    updatePrice(); }
```

```
private void updatePrice()
{   if (controller != null)
    setPrice(super.getPrice() + controller.getPrice()); }
```

```
public Controller getController()
{   return controller; }
```

```
public void setController(Controller controller)
{   this.controller = controller;
    updatePrice(); }
```

```
public boolean isConnected() {   return isConnected;}
public void connect() {   isConnected = true;}
public void disconnect() {   isConnected = false;}
public String getSoftwareVersion(){return softwareVersion;}
public void updateSoftware(String newVersion)
{
    this.softwareVersion = newVersion;
    System.out.println("Software updated to version: " + newVersion);}
```

```
@Override public void activateFeatures()
{   if (isConnected)
    System.out.println("Activating smart features for "
+ getName() + ": Online mode enabled with " +
controller.getName());
```

```
else
    System.out.println("Activating smart features for "
+ getName() + ": Please connect to enable online
features with controller " + controller.getName()); }
```

```
@Override public String toString()
{   return super.toString() + " - Smart Features
[Connected: " + isConnected + ", Software Version: " +
softwareVersion + "] - " + controller.toString(); }
}
```

main

```
public class Main
{
    public static void main(String[] args)
    {
        // Create some Toy objects
        Toy toy1 = new Toy("Dog", 25.0);

        Controller controller1 = new Controller("Arduino", 15.4);
        Controller controller2 = new Controller("RasberyPie", 134.5);

        SmartToy smartToy1 = new SmartToy("Smart Car", 25.2, controller1);
        SmartToy smartToy2 = new SmartToy("Smart Robot", 40.5, controller2);

        // Create ArrayLists to hold the different objects
        ArrayList<Object> toys = new ArrayList<Object>();

        toys.add(toy1);
        toys.add(smartToy1);
        toys.add(smartToy2);
        toys.add(controller1);
        toys.add(controller2);
```

המשך main

```
System.out.println("All Objects:");
for (Object obj : toys)
    System.out.println(obj);

// Print only SmartToys
System.out.println("\nSmart Toys:");
for (Object obj : toys)
    if (obj instanceof SmartToy)
        System.out.println(obj);

}
}//end of class
```

All Objects:

```
Toy: Dog - ID: 1 - Price: $25.0
Smart Toy: Smart Car - ID: 2 - Price: $40.6 - Controller: Arduino - Price: $15.4
Smart Toy: Smart Robot - ID: 3 - Price: $175.0 - Controller: RasberyPie - Price: $134.5
Controller: Arduino - Price: $15.4
Controller: RasberyPie - Price: $134.5
```

Smart Toys:

```
Smart Toy: Smart Car - ID: 2 - Price: $40.6 - Controller: Arduino - Price: $15.4
Smart Toy: Smart Robot - ID: 3 - Price: $175.0 - Controller: RasberyPie - Price: $134.5
```

אב קדמון של כולם - Object

במחלקה Object מוגדרות מетодות הבאות:

- `toString()`
- `hashCode()`
- `equals(Object obj)`
- `finalize()`
- `getClass()`
- `clone()`
- `wait(), notify() notifyAll()`

- בJava כל מחלקה יורשת ממחלקה **Object** אב קדמון
- לכן כל המחלקות בהכרח דורסות ממשות או מקבלות את методות של מחלקת `.Object`
- לכן כמעט בכל המחלקות שimplement ממשות `toString`

@Override

דרישה overriding

- להבדיל מ`C#` כל המethodות בJAVA מדומות `virtual`
- ברירת מחדל
- לכן כל מетодה תמיד ניתנת לדרישה `overriding`.
- אם באמת עשינו דרישת למетодה במחקלה נוצרת איזי מקובל לשים מילה

@Override

```
public class Toy
{
    private static int nextId = 1;

    private String name;
    private int id;
    private double price;
    ...
    @Override public boolean equals(Object obj)
    {
        ...
    }
}
```

שאלה



יש להוסיף מתודה
?Toy.equals
האם להתייחס?id?

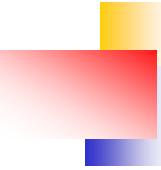
```
public class Toy
{
// Static variable to keep track of the next ID to be assigned
    private static int nextId = 1;

    private String name;
    private int id;
    private double price;
    ...

@Override
public boolean equals(Object obj)
{
    if (obj == null || !(obj instanceof Toy))
        return false;
    Toy other = (Toy)obj;
    return this.name.equals(other.getName()) && this.price == other.getPrice();
}
// Override the finalize method
@Override protected void finalize() throws Throwable
{ System.out.println("Finalizing " + this); }
```

יש להוסיף מетодות
- finalize ו Toy.equals

מחלקה אבסטרקטית



- מחלקת אבסטרקטית **abstract** מכילה מетодות אבסטרקטיות ויכולת להכיל מетодות רגילים.
- לא ניתן ליצור אובייקטים מסווג מחלוקת זו.
- **תחברה:**

abstract class A

{

```
public abstract void method();
//other methods;
```

}

מחלקה מופשטת

```
public abstract class Man
{
    public abstract void sayHello();
    public void introduce() {
        System.out.println("I am a Man.");
    }
}

class Main
{
    public static void main(String[] args)
    {
        Man m = new Man(); //error
        m.introduce();
    }
}
```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Cannot instantiate the type Man1

at Lecture1/Student.Main.main([Main.java:6](#))

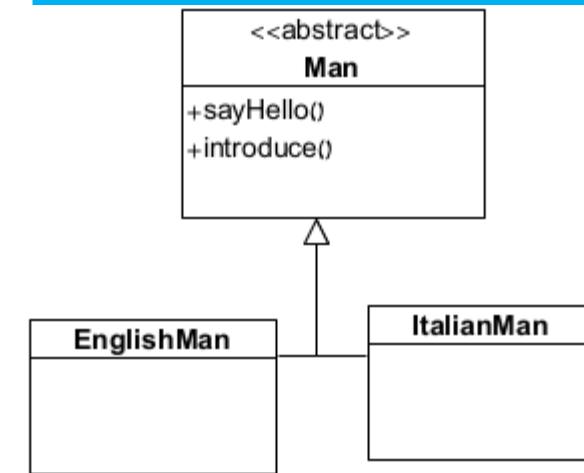
מחלקה מופשטת - 2

```
public abstract class Man
{
    public abstract void sayHello();
    public void introduce()
    {   System.out.println("I am a Man.");   }
}
```

```
public class EnglishMan extends Man
{
    public void sayHello()
    { System.out.println("Good morning!"); }
}
```

```
public class ItalianMan extends Man
{
    public void sayHello()
    { System.out.println("Buongiorno!"); }

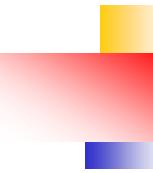
    public void introduce()
    { System.out.println("I am an Italian Man."); }
}
```



```
class Main
{
    public static void main(String[] args)
    {
        //polymorphism
        Man im = new ItalianMan ();
        im.sayHello();
        im.introduce();
    }
}
```

Buongiorno!
I am an Italian Man.

Final class



מחלקה שלא ניתן לרשת ממנה

final : Cannot be extended further

JAVA

```
public class Base
{
    protected void Foo() {}

public final class TypeResolver extends Base
{
    @Override
    protected final void Foo() {}
}
```

C#

```
public class Base
{
    protected virtual void Foo () {}

public sealed class TypeResolver : Base
{
    sealed protected override void Foo () {}
}
```

ממשק Interface



- ממשק זו מחלוקת שכל המתודות שלה אבסטרקטיות.
- מקובל לתת שם שמתחיל באות I
- **תחביר:**

```
interface IName {  
    public void method1(); // interface method  
    public void method2(); // does not have a body  
}
```

- יותר מאוחר מחלוקת ממשית implements ממשק
- class C implements Iname{...}

המשך הספר

בעקבות ההצלחה של הצעדים החכמים והבקרים, הchnerות החליטה להוסיף ממשק אינטראקטיבי חדש לצעדים בשם **IPlayble**.

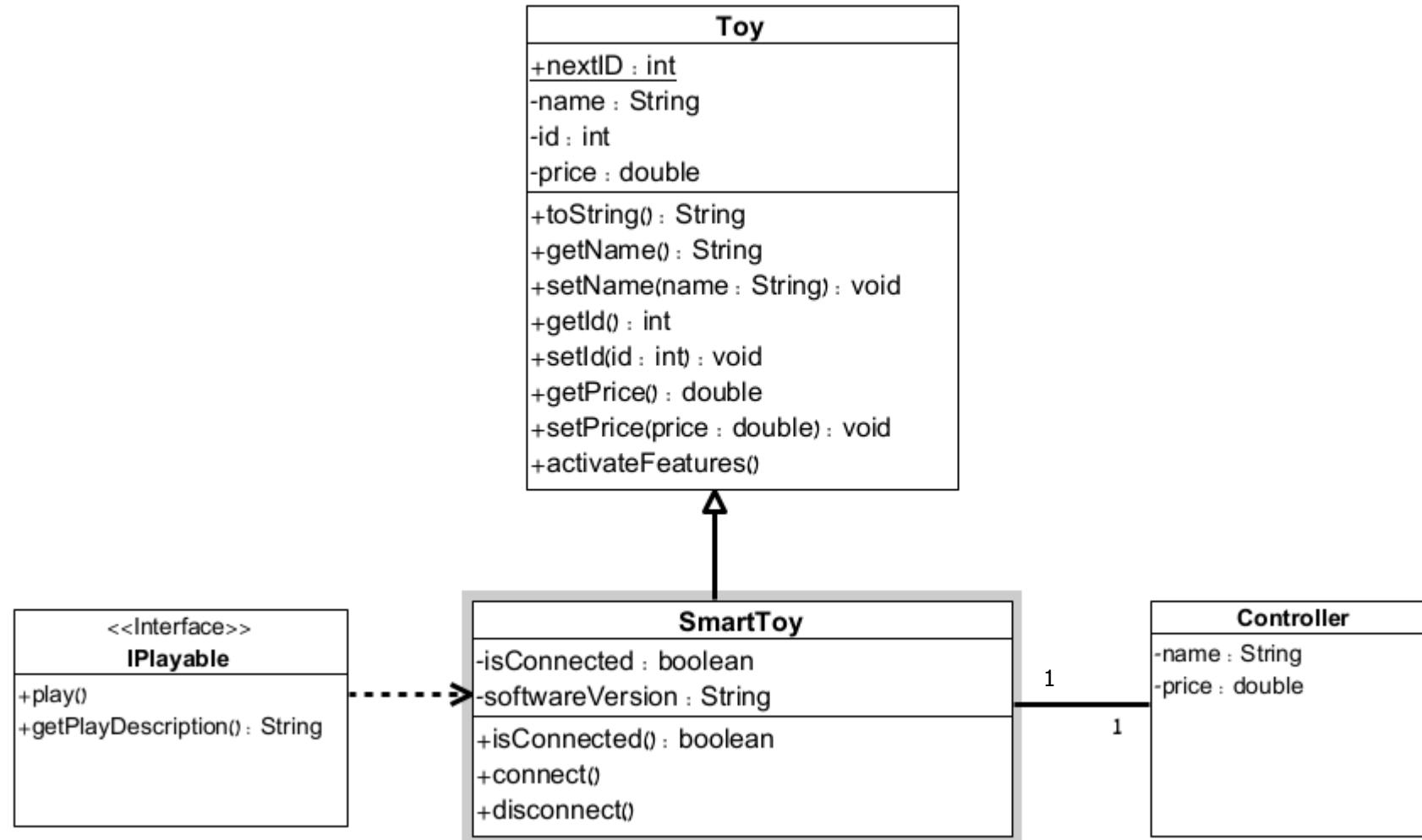
המשק מאפשר לכל צעду חכם לתאר את חווית המשחק שהוא מציע ולתת לילדים חוויה מתקדמת ו互動יבית עוד יותר.

המשק כולל שתי מתודות:

- מפעילה את הצעду ומציגה את שם הצעду ואת שם הבקר, כך שהילדים יכולים להבין איך הצעду נשלט.
- מחזירה תיאור קצר על חווית המשחק שהצעду מציע.

כasher אובייקט **SmartToy** נוסף לרשימה הצעדים של הchnerות, ניתן לבדוק האם הוא מימוש את **IPlayble** ולהפעיל את מתודת **play** כך שהילד יוכל להתחיל לשחק מיד. התהליך הזה גם מאפשר להציג תיאורים שונים של הצעדים בחנות – לפי סוג הצעду והבקר שהוא כולל.

דיאגרמת מחלקות



IMPLEMENTATION

```
public interface IPlayable
{
    void play();
    String getPlayDescription();
}
```

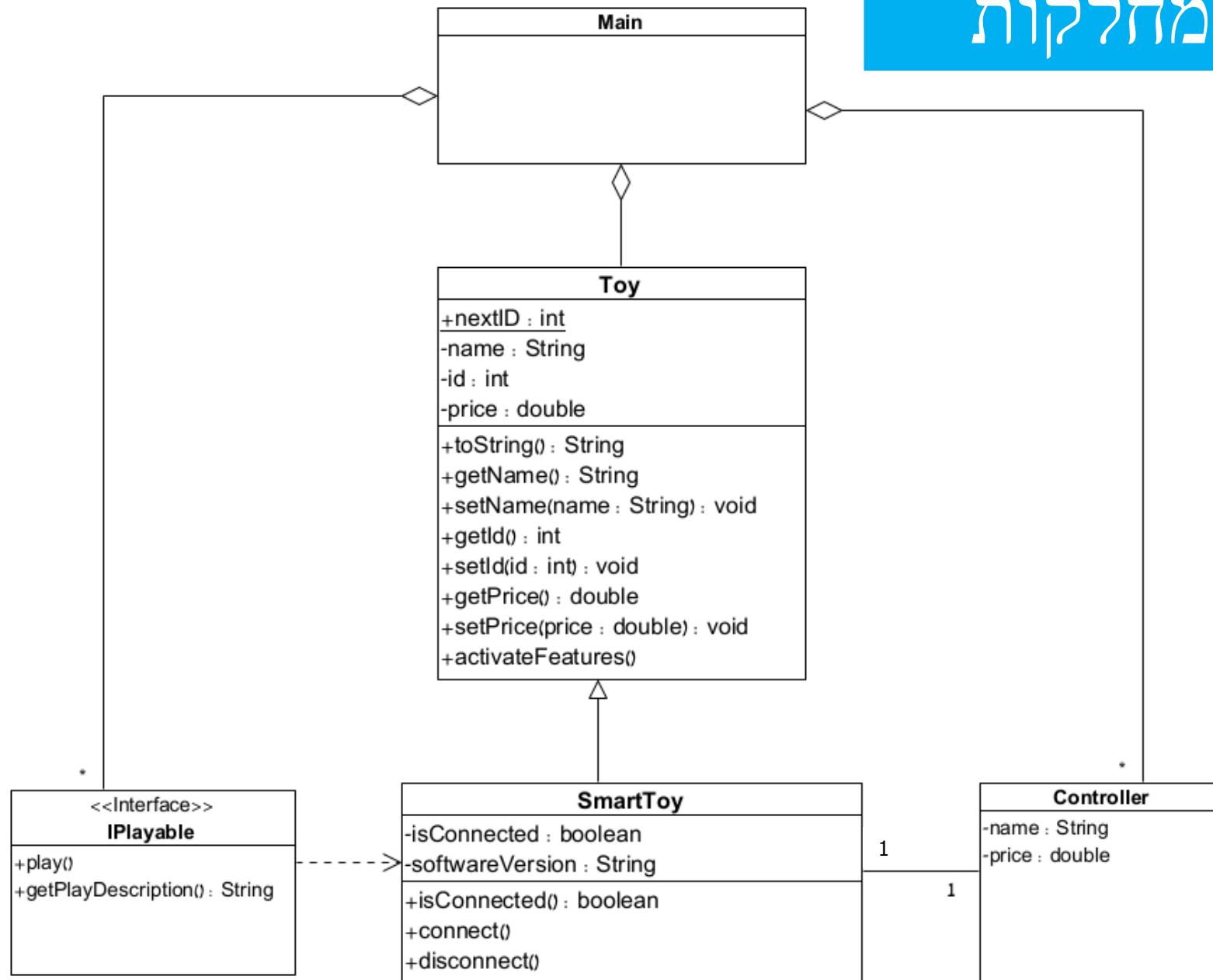
```
public class SmartToy extends Toy implements IPlayable
{
    private Controller controller;
    private boolean isConnected;
    private String softwareVersion;
    .....
Like before.....
.....
```

Like before.....

```
//play implementation
@Override
public void play()
{
    System.out.println("Playing with " + getName() +
" using " + controller.getName() + ".");
}
```

```
//getPlayDescription implementation
@Override
public String getPlayDescription()
{
    return "This smart toy is controlled by " +
controller.getName() + ".";
}
```

דיאגרמת מחלקות



main

```
import java.util.ArrayList;

public class Main
{
    public static void main(String[] args)
    {
        // Create some Toy objects
        Toy toy1 = new Toy("Dog", 25.0);

        Controller controller1 = new Controller("Arduino", 15.4);
        Controller controller2 = new Controller("RasberyPie", 134.5);

        SmartToy smartToy1 = new SmartToy("Smart Car", 25.2, controller1);
        SmartToy smartToy2 = new SmartToy("Smart Robot", 40.5, controller2);

        // Create ArrayLists to hold the different objects
        ArrayList<Object> toys = new ArrayList<Object>();

        toys.add(toy1);
        toys.add(smartToy1);
        toys.add(smartToy2);
        toys.add(controller1);
        toys.add(controller2);
```

```
// Print only SmartToys
System.out.println("\nSmart Toys:");
for (Object obj : toys)
    if (obj instanceof IPlayable)
        ((IPlayable) obj).play();
}
}//end of class
```

Smart Toys:

Playing with Smart Car using Arduino.

Playing with Smart Robot using RasberryPie.

Instance Initializer

```
public class Example
{
    private final String _name;
    private final int _type;

    //instance initializer
    {
        _name = "Example";
    }

    //constructor
    public Example ( int type )
    { _type = type; }
}
```

- ניתן לבצע **איתחול מופע** לפני כניסה לבנאי.
- אין שם.

- בлок {} ממוקם בתוך המחלקה אבל אף לא שייך לאף
מתודה או בנאי.

- לא ניתן לקרוא בצורה מפורשת.

- כל איתחולי המופעים מתרחשים מלמעלה כלפי מטה בקוד
לפני קריאה לבנאי.

- **חשוב:** אם מדובר במחלקה נגזרת או קודם מתבצעת קריאה
לבנאי האב(super) וואז איתחולי מופע ורק לאחר מכן
בנאי הבן.



סוגי חריגות



- ישנו חיריגות מובנות/מודכנות (**built-in**) ויש חריגות משתמש.
 - חריגות מובנות
 - **בדיקות checked** חריגת זמן קומpileציה
 - דוגמה IOException, SocketException
 - **לא נבדקות unchecked** חריגת זמן הריצה
 - דוגמה IndexOutOfBoundsException, ArithmeticException
 - **חריגות משתמש** - חריגה שמשתמש יצר
- ```
class NewException extends Exception{ ...}
```

# Throwable

- ב-Java קיימת מחלקה `Throwable` שמייצגת מצב שגוי
- מחלקות `Error` ו- `Exception` יורשות ממנה.
- **מנגנון :**
- כאשר מתרחשת שגיאת ריצה נוצר אובייקט מסווג חריגה והוא "נזרק" `throw`
- בנוסף "דגל השגיאות" נדלק ומסמן שיש בעיה וכל עוד לא נטפל בה לא ניתן להמשיך בתוכנית הרגילה
- התוכנית יכולה "لتפוס" `catch` את החריגה ולטפל בה.

# main

```
public class mainClass
{
 public static void main(String[] args)
 {
 try {
 // Create some Toy objects
 Toy toy1 = new Toy("Dog", 25.0);
 System.out.println(toy1);
 Toy toy2 = new Toy("Dog", -45.2);
 System.out.println(toy2);
 }
 catch (IllegalArgumentException e)
 {
 System.out.println("Exception caught: " + e.getMessage());
 }
 } //end of main
} //end of mainClass
```

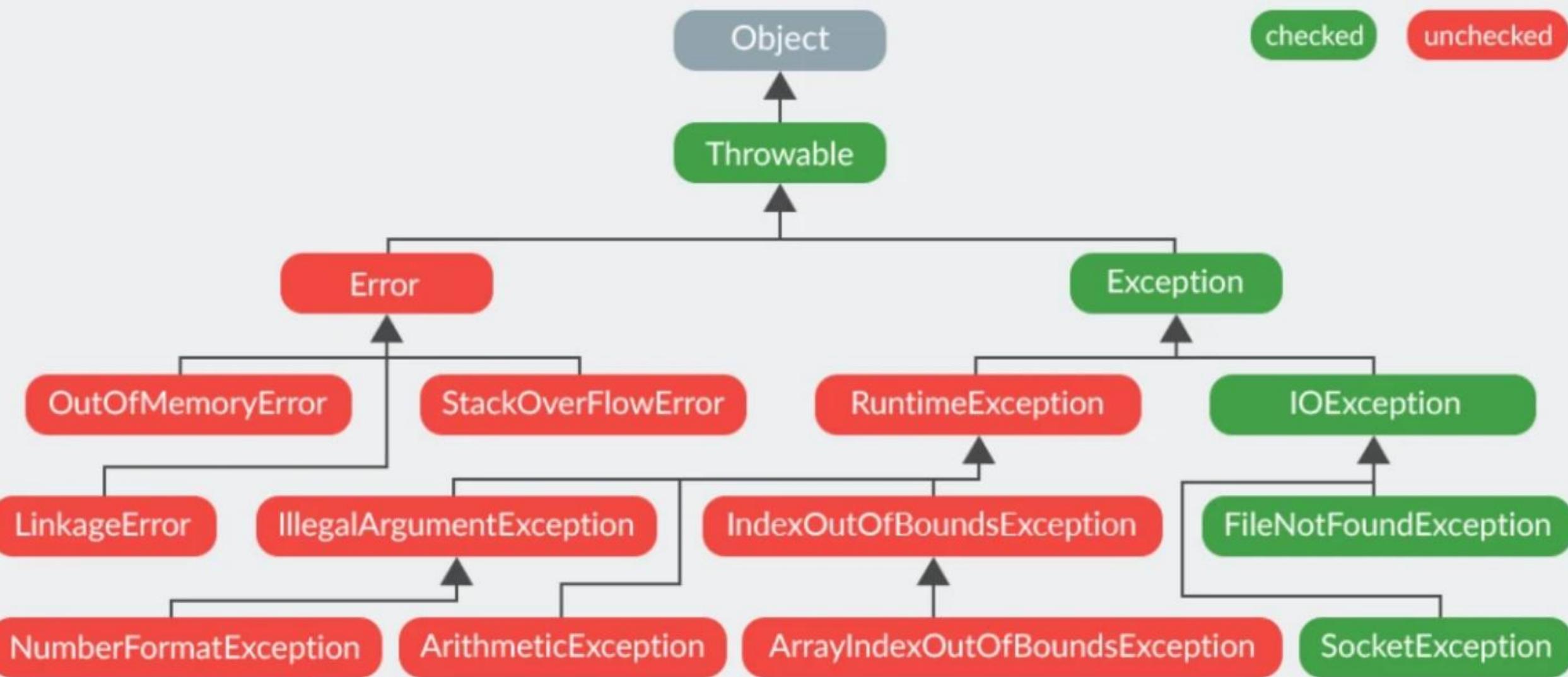
```
Toy: Dog - ID: 1 - Price: $25.0
Exception caught: Price cannot be negative.
```

# main

```
public class MainClass
{
 public static void main(String[] args)
 {
 // Create some Toy objects
 Toy toy1 = new Toy("Dog", 25.0);
 System.out.println(toy1);
 Toy toy2 = new Toy("Dog", -45.2);
 System.out.println(toy2);
 } //end of main
} //end of mainClass
```

כתיבה לא מסודרת  
ללא try catch

```
<terminated> app [Java Application] C:\Users\paral\Downloads\eclipse-java-2020-12-R-win32-x86_64\ec
Exception in thread "main" Toy: Dog - ID: 1 - Price: $25.0
java.lang.IllegalArgumentException: Price cannot be negative.
 at Toy.setPrice(Toy.java:82)
 at Toy.<init>(Toy.java:34)
 at app.main(app.java:14)
```



# Checked Vs Unchecked



- כל שגיאה שיורשת מהמחלקות `Error` או `RuntimeException` מוגדרת **כלא נבדקות**.
- כל שאר החריגות מוגדרות **כנבדקות**.
- חריגות נבדקות כל מトודה צריכה:
  - או לceptors .. catch את החריגה בגוף המטודה ולטפל בה .1
  - או להציג בחתימה שהיא לא תופסת את החריגה throws .2
- **דוגמה:**

```
public void fun() throws Exception1, Exception2
```

## דוגמה חריגת משתמש user exception

```
class InvalidPaymentException extends Exception
{
 public InvalidPaymentException(String message)
 {
 super(message);
 }
}
```

# חריגות נפוצה

- NullPointerException - 70%
- NumberFormatException – 55%
- IllegalArgumentException – 50%
- RuntimeException – 23%
- NoSuchMethodException – 16%
- ClassCastException – 15%
- Exception – 15%
- ArrayIndexOutOfBoundsException
- FileNotFoundException

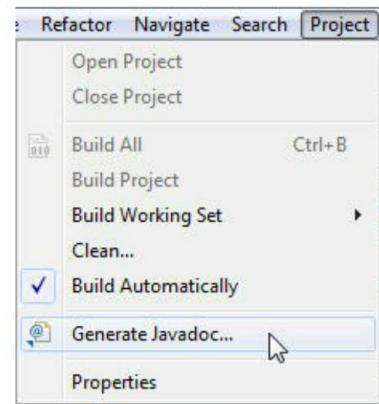
## חורשה וחריגות

```
public class A
{
 public void fun() throws IOException,
InterruptedException
 {...}
}

public class B extends A
{
 public void fun() throws EOFException,
InterruptedException, ClassNotFoundException
 {...}
}
```

- המетодה הדרשת **לא יכולה** להציג על חריגות שלא הוצהרו על ידי המетодה המקורית או שלא ירשות מחריגות כאלה
- IOException ירשת מ EOFException  
ולכן **מותר** ClassNotFoundException לא מוגדרת אצל האב ולכן  **אסור**

# javadoc



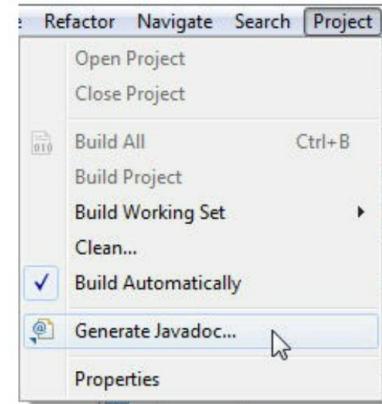
• **הтиיעוד עברו קובץ תקין מייצר קובץ HTML של המחלקה.**

```
/** * Calculates the sum of two numbers.
*
* @param a the first number
* @param b the second number
* @return the sum of a and b
*/

public int sum(int a, int b)
{ return a + b; }
```

• **צורת כתיבה מתחילה ב \*/ כל שורה \* בסוף \*/.**

• **דוגמה :**



/\*  
 \* The {@code SmartToy} class provides a smart toy.  
 \* It provides the price of the toy.  
 \* The price is the sum of the price of the toy and the price of the controller.  
 \*/  
**public class** SmartToy  
 extends Toy  
 implements IPlayable  
  
**private** This class is intended to be used in applications where managing smart toy and controller data is required.  
  
Version:  
1.2  
\* Cons Author:  
\* The Rina Zviel  
\*  
\* @param Constructor Summary  
\* @param Constructors

Questions  
? Answers  
?