אובייקטים ומחלקות חלק 2
שיטות

- חתימה של שיטה.
- העברת פרמטרים לשיטה.
- החזרת ערך משיטה.

שיטות

- מחלקה יכולה להכיל שיטות.
- שיטה מוסיפה פונקציונליות לאובייקט – מה שהאובייקט יכול לעשות.

קובע מי רשאי להפעיל את השיטה.

שם שישמש לזיהוי השיטה. מתחיל באות קטנה, ורצוי שיתאר מה השיטה עושה.

```
מאפיין גישה   ערך חוזר   שם השיטה(רשימת פרמטרים)
{
   גוף השיטה
}
```

סוג הנתון שהשיטה מחזירה – int, double וכו'. אם השיטה לא מחזירה כלום, כותבים void.

נתונים שהשיטה מקבלת ממי שמפעיל אותה. אם השיטה לא מקבלת נתונים, לא רושמים כלום בין הסוגריים.

חתימת השיטה – Method's Signature

```
public class Person
{
   public String name;
   public int age;
   public void sayHello()
   {
      System.out.println("Hello");
   }
}
```

```
public static void main(String[] args) {
  Person p = new Person();
  p.sayHello();
  …
}
```

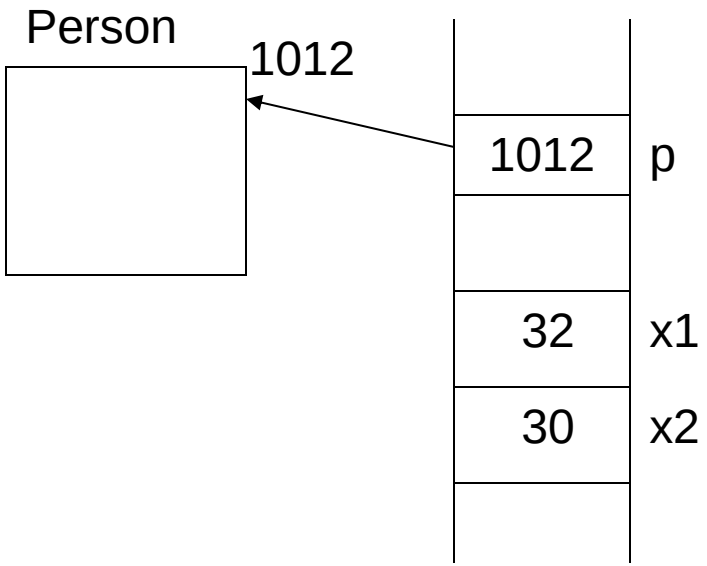
שיטה שמקבלת פרמטרים

- נוסיף למחלקה Person שיטה בשם parentsAges.
- השיטה תקבל את גילאי ההורים, ותדפיס על המסך בכמה גדולים ההורים מהאדם.
- בניגוד לשיטה sayHello שראינו קודם, שיטה זו צריכה לקבל מידע חיצוני לפני שהיא פועלת (במקרה זה, גיל ההורים). מידע זה מועבר ב**פרמטרים**.

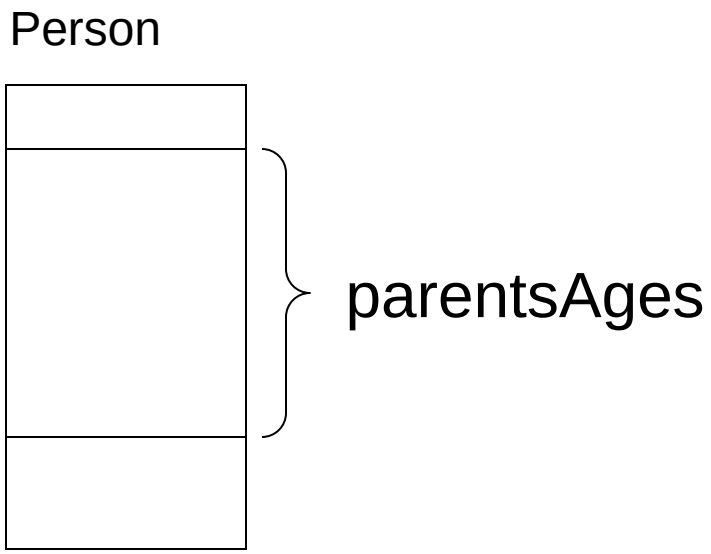
```
public class Person
{
   public String name;
   public int age;
   public void parentsAges(int father, int mother)
   {
      int fDiff, mDiff;
      fDiff = father – age;
      mDiff = mother – age;
      System.out.println(fDiff + “, “ + mDiff);
   }
}
```

הפרמטרים של השיטה – השיטה מקבלת שני פרמטרים מסוג int.
כלומר, מי שמפעיל את השיטה מחוייב לספק לה שני מספרים בזמן ההפעלה.

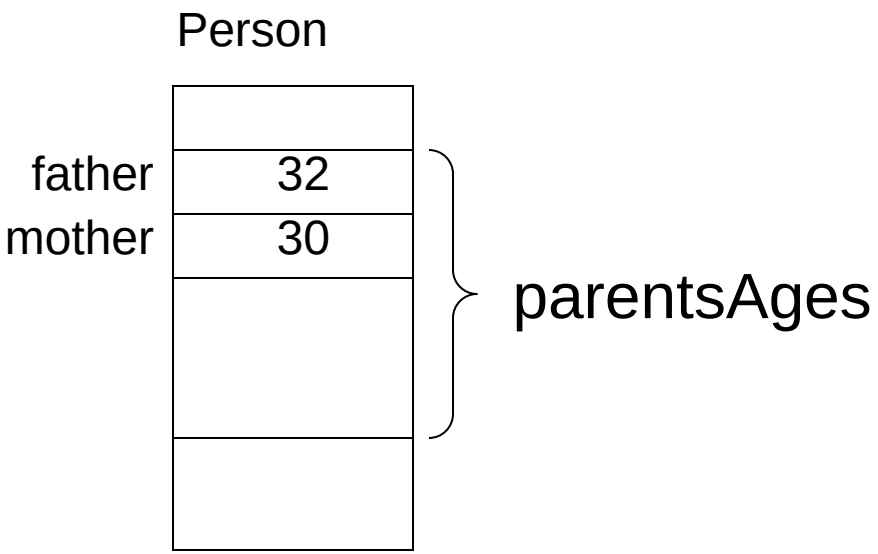
```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}
```



```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}
```

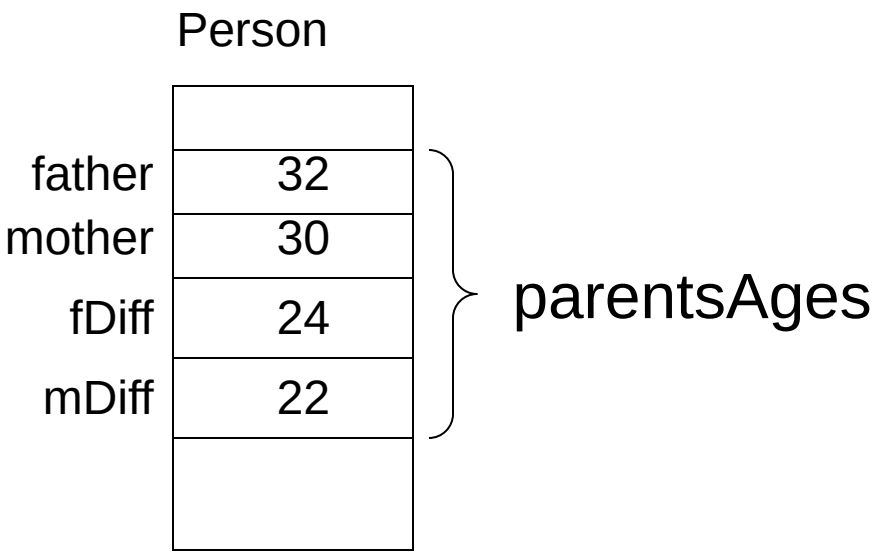


```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}
```



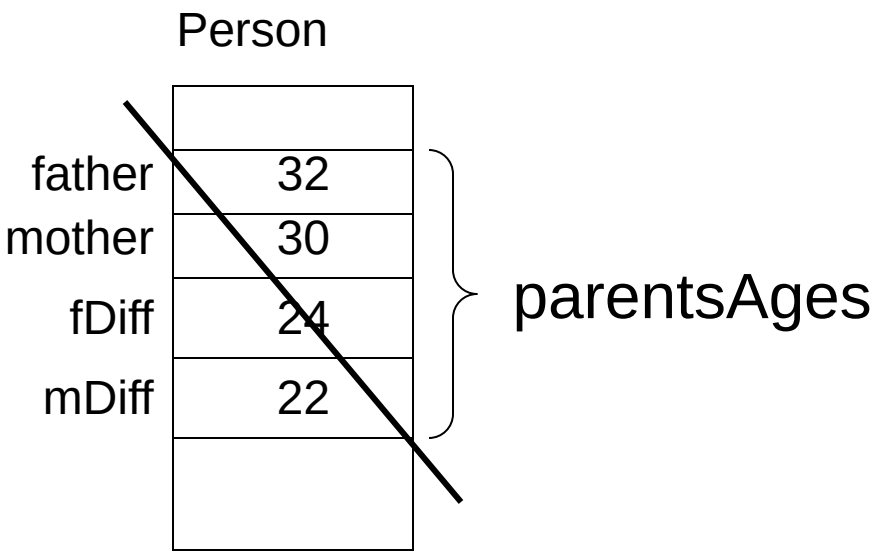

```
public void parentsAges(int father, int mother)
  {
     int fDiff, mDiff;
     fDiff = father – age;
     mDiff = mother – age;
     System.out.println(fDiff + “, “ + mDiff);
  }
```

```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}
```




```
public void parentsAges(int father, int mother)
  {
     int fDiff, mDiff;
     fDiff = father – age;
     mDiff = mother – age;
     System.out.println(fDiff + “, “ + mDiff);
  }
```

```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}
```




```
public void parentsAges(int father, int mother)
  {
     int fDiff, mDiff;
     fDiff = father – age;
     mDiff = mother – age;
     System.out.println(fDiff + “, “ + mDiff);
  }
```

השלבים בקריאה לשיטה

- בקריאה לשיטה נפתח מרחב זיכרון עצמאי בתוכו תרוץ השיטה.
- לכל פרמטר ב**חתימת השיטה** מוקצה משתנה במרחב הזיכרון של השיטה.
- לכל פרמטר **מועתק** הערך שמופיע בקריאה לשיטה.
- בסיום השיטה נמחק כל מרחב הזיכרון שלה, והשליטה חוזרת חזרה למי שקרא לשיטה.

```
  public static void main(String[] args)
  {
     int x1 = 32, x2 = 30;
     Person p = new Person();
     p.age = 8;
     p.parentsAges(x1, x2);
  }
public void parentsAges(int father, int mother)
  {
     int fDiff, mDiff;
     fDiff = x1– age;
     mDiff = x2– age;
     System.out.println(fDiff + “, “ + mDiff);
  }
```

שגיאת קומפילציה, x1 ו-x2 לא מוגדרים בשיטה

```
public static void main(String[] args)
{
    int x1 = 32, x2 = 30;
    Person p = new Person();
    p.age = 8;
    p.parentsAges(x1, x2);
    System.out.println(fDiff);
}
public void parentsAges(int father, int mother)
{
    int fDiff, mDiff;
    fDiff = father– age;
    mDiff = mother– age;
    System.out.println(fDiff + “, “ + mDiff);
}
```

שגיאת קומפילציה, fDiff
לא מוגדר ב-main

```
public static void main(String[] args)
{
    int x1 = 32, x2 = 30;
    Person p = new Person();
    p.age = 8;
    p.parentsAges(x1, x2);
    System.out.println(x1);
}
public void parentsAges(int father, int mother)
{
    int fDiff, mDiff;
    fDiff = father– age;
    mDiff = mother– age;
    System.out.println(fDiff + “, “ + mDiff);
    father = 90;
}
```

הפלט יהיה 32, השינוי שביצעה השיטה לא התבצע במקור!

העברת פרמטרים By Value

- בג'אווה הפרמטרים לשיטות עוברים תמיד By Value, כלומר, השיטה מקבלת **עותק** של הפרמטר, ולא את הפרמטר המקורי.
- המסקנה היא שכל שינוי שעושה השיטה על הפרמטרים שהיא מקבלת הוא שינוי שנעשה על עותק, וכשהשיטה נגמרת, המשתנה המקורי נותר ללא שינוי.

```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1, x2);
}

public void parentsAges(int x1, int x2)
{
   int fDiff, mDiff;
   fDiff = x1– age;
   mDiff = x2– age;
   System.out.println(fDiff + “, “ + mDiff);
}
```

חוקי – לשיטה יהיו משתנים מקומיים בשמות אלו

```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(x1);
}

public void parentsAges(int father, int mother)
{
   int fDiff, mDiff;
   fDiff = father– age;
   mDiff = mother– age;
   System.out.println(fDiff + “, “ + mDiff);
}
```

שגיאת קומפילציה –
חתימת השיטה מחייבת
להעביר שני פרמטרים

```
public static void main(String[] args)
{
   int x1 = 32, x2 = 30;
   Person p = new Person();
   p.age = 8;
   p.parentsAges(3.4, 30);
}
public void parentsAges(int father, int mother)
{
   int fDiff, mDiff;
   fDiff = father– age;
   mDiff = mother– age;
   System.out.println(fDiff + “, “ + mDiff);
}
```

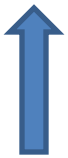
שגיאת קומפילציה – חתימת השיטה מחייבת להעביר שני פרמטרים מסוג int

החזרת ערך משיטה

- שיטה יכולה להחזיר ערך בסיומה.
- שיטה יכולה להחזיר לכל היותר ערך אחד, באמצעות הפקודה return.
- פקודה זו עוצרת את השיטה, ובנוסף יכולה גם להחזיר מספר אחד למי שקרא לשיטה.
- למשל, נוסיף למחלקה Person שיטה שמחזירה כמה שנים נשארו לבן אדם עד גיל הפרישה.

```
public class Person
{
   public String name;
   public int age;
   public int retire()
   {
      int res = 64 – age;
      return res;
   }
}
```

```
public static void main(String[] args)
{
   int res;
   Person p = new Person();
   p.setAge = 30;
   res = p.retire();
   System.out.println(p.retire());
   p.retire();
}
```



חוקי, אבל כיוון שהערך החוזר לא מוצב
בשום מקום הוא "נעלם"

```
public class Person
{
   public String name;
   public int age;
   public void retire()
   {
      int res = 64 – age;
      return res;
   }
}
```

שגיאת קומפילציה – שיטה שמסומנת כ-void לא יכולה להחזיר ערך

```
public class Person
{
   public String name;
   public int age;
   public int retire()
   {
      return 9.67;
   }
}
```

שגיאת קומפילציה – שיטה שמסומנת כ-int לא יכולה להחזיר double.

תרגיל

- כתבו מחלקה בשם Book המייצגת ספר. הוסיפו למחלקה תכונה של שם הספר ושם הסופר.
- הוסיפו למחלקה שיטה בשם printDetails. השיטה תדפיס על המסך את שם הספר ושם הסופר, מופרדים בפסיק.
- כתבו main במחלקה אחרת וצרו בו אובייקטים מסוג Book. הפעילו את השיטה וראו איך היא עובדת.