פרק 3. חיית מחמד ודג – מחלקות Java.

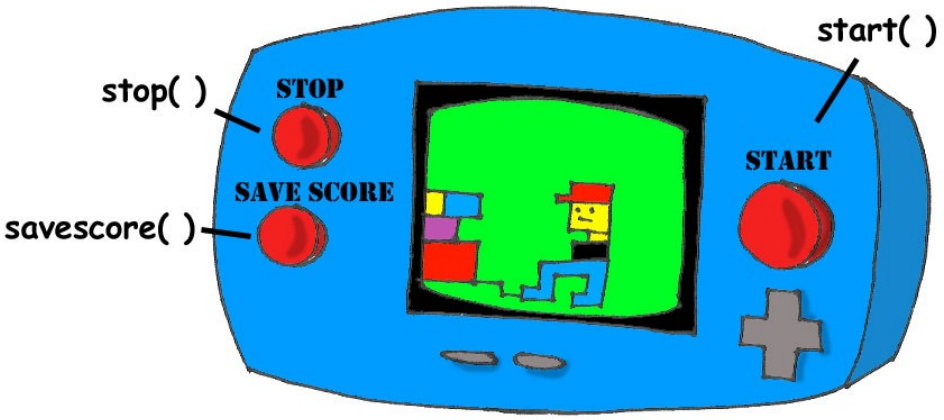
Translation from English by Nataly Shnaidman, Jerusalem

תוכנות Java מורכבות ממחלקות (classes) שמציגות אובייקטים מעולם אמיתי. אפילו שלאנשים יש דעות שונות איך לכתוב תוכניות, רובם מסכימים שהכי טוב לעשות את זה בסגנון שניקרא Object-oriented. זה אומר שמתכנת טוב מתחיל עם החלתה איזה אובייקטים התוכנית תכלול ואיזה מחלקות Java ייצגו אותם. רק אחרי שהעבודה הזאת הסתימה הם מתחילים לכתוב קוד ב- Java.

מחלקות ואובייקטים

למחלקות ב-Java ישנם שיטות(methods) ותכונות ((attributes. שיטות אלה מגדירות פעילויות שהמחלקות יכולות לבצעם. התכונות מתארות את המחלקה.

באו נצור ונדון במחלקה בשם VideoGame.
למחלקה הזאות קיימות מספר שיטות שמאפיינות מה האובייקטים של מחלקה יכולים לעשות: להתחיל משחק, לעצור אותו, לשמור תוצאות וכו'. למחלקה הזאת ישנם גם כמה תכונות או מאפיינים: מחיר, צבע מסך, כמות שלטים מרוחקים ואחרים.



בשפת Java המחלקה הזאת יכולה להראות כך:

```
} classs VideoGame
;String color
;Int price
} ()void start
  {
}()void stop
 {
} (void saveScore(String playerName,int score
```

```
{
 {
```

המחלקה שלנו תהיה דומה למחלקות אחרות שמתארות משחקי וידאו –
לכולם קיימות מסכים במידות שונות וצבעים שונים, כולם מבצעים
פעולות דומות ולכולם יש מחיר.

אנו יכולים להיות יותר ספציפיים ולייצר מחלקה אחרת בשם GameBoyAdvance
היא גם שייכת למשפחת משחקי וידיאו אבל יש לה מאפיינים ספציפיים רק לדגם
GameBoyAdvance, למשל סוג קסטה.

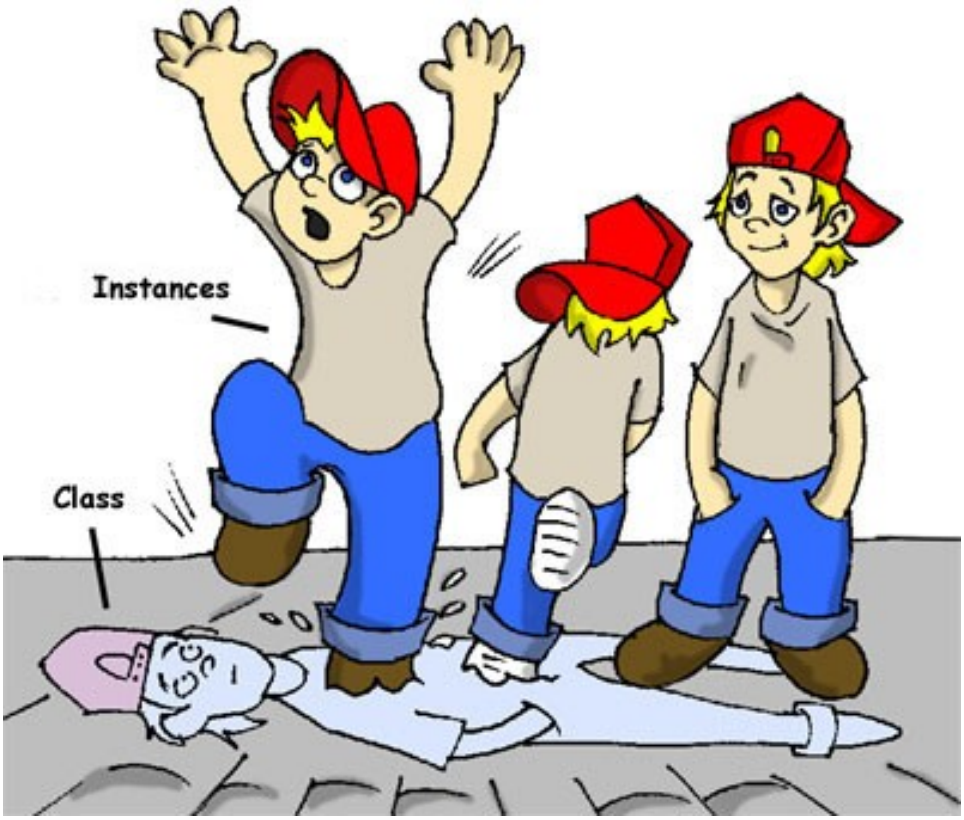
```
} Class GameBoyAdvance
;String cartridgeType
;Int ScreenWidth

} ()void startGame
{
     } ()void stopGame
{
{
```

בדוגמה הזאת מחלקה GameBoyAdvance מגדירה שתי תכונות – cartrhdgeType
ו- screenWidth ושתי שיטות- startGame() ו- stopGame(). שיטות האלה עדיין
לא יכולות לבצע שום פעולה כי חסר קוד בין הסוגריים מסולסלים.

בנוסף למילה מחלקה (class) צריך להתרגל למשמעות חדשה של המילה אובייקט.
משפט "ליצור מופע של אובייקט" הכוונה ליצירת העתק של אובייקט הזה
בזיכרון של המחשב לפי הגדרות של מחלקתו.

הגדרת המפעל של GameBoyAdvance מתייחסת למשחק ראלי כמו
מחלקת Java מתייחסת למופע שלו בזיכרון. תהליך של בניית משחק ראלי
שמבוסס על הגדרה הזאת במפעל של משחק דומה ליצירת מופע של אובייקט
GameBoy ב-Java.



במקרים רבים התוכנית יכולה להשתמש במחלקת Java רק אחרי יצירת המופע שלה.
מוכרים מיצרים אלפי העתקים של משחקים מבוססים על אותה הגדרה. אף על פי ש
ההעתקים האלה מייצגים אותה מחלקה לתכונות שלהם יכולים להיות ערכים שונים –
למשל צבע שונה וכו'. במילים אחרות תוכנית יכולה לייצר מספר רב של מופעי
אובייקטים GameBoyAdvance .

סוגי משתנים

משתני Java מציגות תכונות של מחלקות, ארגומנטים של שיטות
או יכולים להשתמש בהם בתוך שיטה לשמירת נתונים לזמן קצר.
רק אחרי הגדרת משתנים אפשר יהיה להשתמש בהם.

זוכרים את המשוואה ?y=x+2 ב- Javaאתם צריכים קודם להגדיר משתנים
xו- yבתור משתנים מסוג מספרי integer או double:

```
int x;
int y;
```

שתי השורות הבאות, מראות איך אפשר לתת ערך למשתנים האלה.
אם התוכנה שלהם מקצה ערך 5 ל- משטנה x, משתנה y יקבל ערך 7.

```
x=5;
y=x+2;
```

ב- Java אפשר לשנות ערך משטנה בדרך לא רגילה.
שתי השורות הבאות משנים ערך משטנה y מחמש לשש.

```
int =5;
```

;++y

בניגוד לשני סימנים "+" ערך משטנה y יגדל ב1-.

אחרי הקטע של הקוד הבא הערך של המשתנה myScore יהיה גם שש.

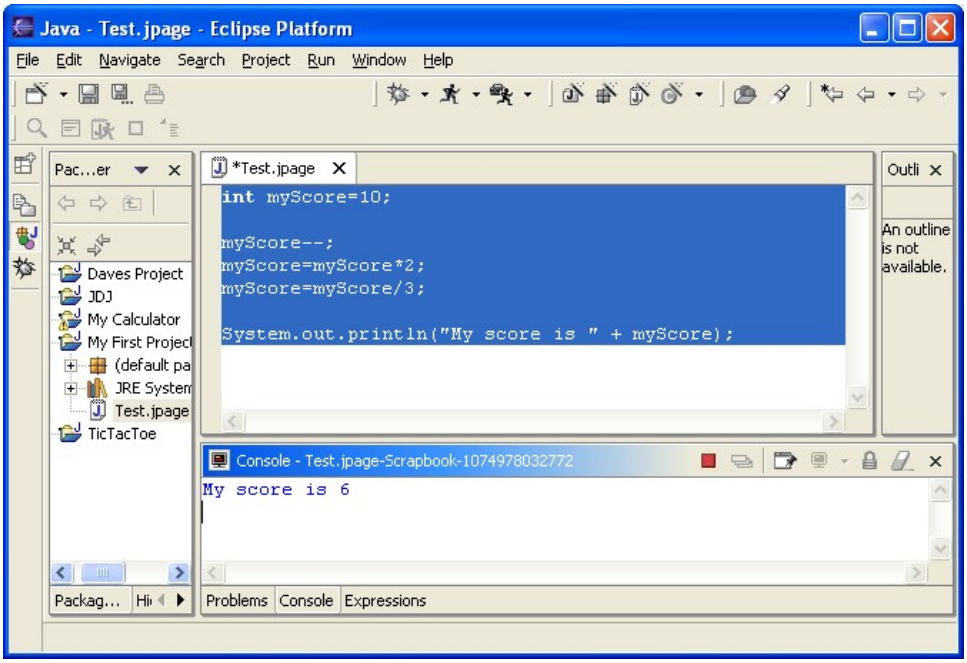
;Int myScore=5
;MyScore=myScore+1

לפעולות של כפל, חלוקה והפחתה אפשר להשתמש באותה דרך:

```
;Int myScore = 10
;--myScore
;myScore = myScore*2
;myScore = myScore/3
  ;(System.out.printl(My score is + myScore
```

מה מדפיס קוד הזה?

ל- Eclipse ישנו תכונה מגניבה שנקראת scrapbook
שמאפשרת לבדוק קטע קוד (כמו שכתוב לעיל) מהר מבלי ליצור מחלקה.
בחר מהתפריט File > New > Scrapbook Page ותקיש מילה
Test כשם קובץ scrapbook שלך.
הכנס חמש שרות קוד שכתוב לעיל, סמן אותם ולחץ על מראה קטנה
ב-toolbar .



כדי לראות את תוצאת החישובים, לחץ על consol tab בתחתית מסך:

My score is 6

בדוגמה זאת הארגומנט של השיטה println() כולל שני חלקים :
מלל "My score is" וערך משתנה myscore ששווה 6.
יצירת מחרוזת מחלקים נקראת concatenation. אפילו שמשתנה myscore הוא נומרי , Java די חכמה כדי להפוך משתנה נומרי למשתנה מסוג – String ואחר כך להצמיד אותה למלל is Myscore.

בוא נראה דרך אחרת לשנות ערך של משתנה:

myscore*=2 זהה ל MyScore=myscore*2
myscore+=2 זהה ל MyScore=myscore+2
myscore-=2 זהה ל MyScore=myscore-2
myscore/=2 זהה ל MyScore=myscore/2

קיימים שמונה סוגי משתנים ב-Java פשוטים או פרימיטיבים,
באיזה סוג לבחור,הדבר תלוי בסוג וגודל הערך שרוצים לשמור במשתנה:

ארבע סוגי משתנים לשמירת ערכים מספריים – byte,short,int,long .

שני סוגי משתנים לערך עם נקודה עשרונית – float ו-double.

סוג אחד לשמירת אות בודדה – char.

וסוג אחד משתנה לוגי שניקרא Boolean ומאפשר רק שני ערכים:
True או False.

אפשר להגדיר ערך התחלתי בזמן הגדרת משתנה וזה נקרא
:Variable initialization

```
 = A'Char grade';
;int chairs = 12
;Boolean playSound = false
;Double nationalIncome = 23863494965745.78
;Float gamePrice = 12.50f
;Long totalCars = 46372836483921
```

בשתי השורות האחרונות f פירושו float ו- l פירושו long.
אם לא נותנים ערך התחלתי למשתנים, Java מבצעת זאת.
משתנה נומרי מקבל 0, משתנה Boolean – false ומשתנה מסוג
Char מקבל קוד מיוחד '\u000' .

קיימת מילת מפתח מיוחדת – final, משתמשים בה בהגדרת משתנים,
אפשר לקבוע ערך למשתנה הזה רק פעם אחת, ואי אפשר לשנות את הערך הזה אחר כך.
קיימות שפות שמשתנים מסוג final נקראים constants. ב-Java
בהגדרת משתנים מסוג final בדרך כלל משתמשים באותיות גדולות:

```
= Washington”  Final String STATE_CAPITAL“
```

בנוסף לסוגי משתנים פרימיטיביים ב-Java אפשר להשתמש במחלקות
להגדרת משתנים. ל כל סוג נתונים פרימיטיביים ישנה במקביל מחלקת
עטיפה, למשל Integer,Double, Boolean וכו'. למחלקות האלה
יש תכונות שימושיות להפוך סוג אחד לסוג אחר.

סוג משתנה char יכול לשמור רק אות אחת, ב-Java יש מחלקה
String לצורך עבודה עם מלל יותר ערוך, למשל:

```
=Smith"String lastName"
```

ב-Java שמות משתנים לא יכולים להתחיל עם מספר או רווחים.

Bit – חלק הכי קטן של מידע שאפשר לשמור בזיכרון. הוא יכול לאחוז 1 או 0.

- Bite כולל 8 ביטים.

Char תופס 2 בתים של זיכרון.

int ו-float תופסים 4 בתים של זיכרון.

משתנים מסוג long ו- double משתמשים ב8- בתים כל אחד.

סוגי משתנים נומריים שמשתמשים ביותר בתים יכולים לאחסן מספר

יותר גדול.

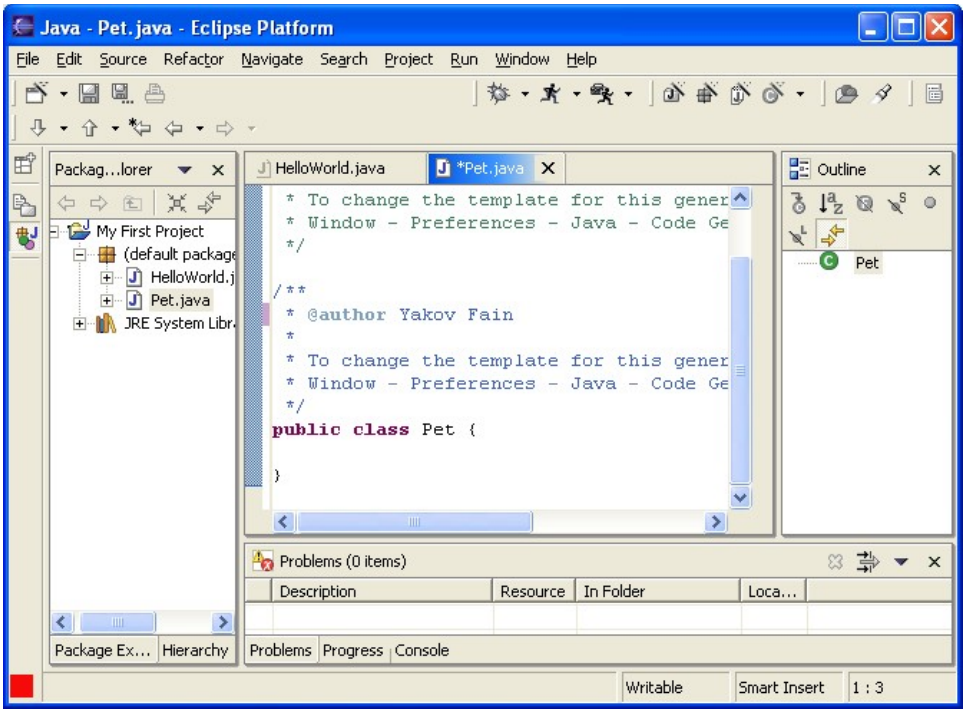
1 KB שווה ל1024- בתים

1 MB שווה ל1024- KB

1 GB שווה ל1024- MB

יצירת חית מחמד

בוא נתכנן וניצור מחלקת חית מחמד. קודם כול צריך להחליט איזה
?פעולות החיה שלנו תעשה: לאכול, לישון ולדבר
אנו נתכנן את הפעולות האלה בתוך שיטות של מחלקת חית מחמד.וכן
ניתן לחיה שלנו מאפיינים כאלה: גיל, גובה, משקל וצבע. נתחיל עם
יצירת מחלקת Java חדשה בשם Pet-ב My First Project
כמו שמתואר בפרק 2, אבל אל תסמן קופסה ליצירת שיטה main() .
המסך שלך צריך להיות כדלהלן:



בשלב זה אנו מוכנים להגדרת תכונות ושיטות במחלקת Pet.
הגדרות מחלקות ושיטות Java נכללות בסוגריים מסולסלות.
כל סוגר מסולסל שנפתח חייב סוגר מסולסל שנסגר.

```
{    }Class Pet
```

להגדרת משתנים לתכונות מחלקה צריך לבחור סוגי משתנים.
אני מציע int - לגיל, float - למשקל וגובה, ו-String - לצבע של החיה.

```
}Class Pet
;Int age
;Float weight
;Float height
;String color
 {
```

צעד הבא – להוסיף תכונות למחלקה הזאת.
לפני הגדרת תכונה, צריך להחליט האם תכונה תקבל ארגומנטים והאם היא תחזיר ערך:

תכונת sleep() תדפיס הודעה "לילה טוב, אראה אותך מחר" – זה לא צריך ארגומנטים ולא יחזור שום ערך.

אותו דבר נכון גם לתכונה eat().זה ידפיס הודעה "אני כל כך רעב ... תן לי לאכול חטיף!".

תכונת say() גם תדפיס הודעה, אבל החיה "תגיד" (תדפיס) את המילה או המשפט שאנחנו ניתן לה.
אנחנו נעביר מילה לתכונה say() ארגומנט של תכונה. התכונה תשתמש בארגומנט הזה לבניית משפט ותחזיר אותו לתוכנה שממנה תכונה נקראה.

גרסה חדשה של מחלקת Pet תיראה כך:

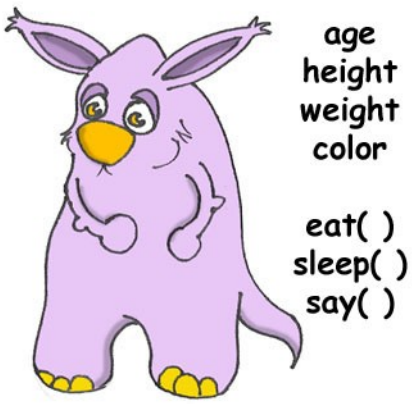
```
} Public class Pet
;Int age
;Float weight
;Float height
;String color

} ()Public void sleep
;("System.out.println("Good night, see you tomorrow
{

} ()public void eat
;("!System.out.println("I am so hungry  ...let me have a snack like nachos
{

}(public String say(String aWord
;String petResponse = "OK!! OK!!"  + aWord
;Return petResponse
{
```

מחלקה הזאת מייצגת יצור ידידותי מעולם אמיתי:



נדבר על הגדרת תכונת sleep() :

()Public void sleep

זה אומר שלתכונה הזאת אפשר לקרא מכל מחלקה אחרת (public), והיא לא מחזירה שום ערך (void). סוגריים ריקות המשמעות שתכונה לא מקבלת ארגומנטים, היא לא צריכה שם ערך מבחוץ כי תמיד מודפס אותו טקסט.

הגדרת תכונה say() נראת כך+

(Public String say(String aWord

כל מחלקה אחרת יכולה לקרוא את התכונה הזאת, אבל היא צריכה להחזיר איזה שהוא טקסט וזה פירוש המילה String לפני שם התכונה. חוץ מזה היא מצפה לאיזה טקסט מבחוץ בגלל ארגומנט String aWord.

איך מחליטים האם תכונה תחזיר ערך? אם תכונה מבצעת איזה פעולות עם נתונים וצריכה להחזיר את התוצאה של הפעולות האלה למחלקה הקוראת, התכונה צריכה להחזיר ערך. אתם יכולים להגיד שלתכונת Pet אין שום מחלקה קוראת! זה נכון, באו ניצור אחת כזאת בשם PetMaster . במחלקה הזאת נמצאת תכונה main() שכוללת קוד להדברות עם מחלקה Pet. תיצרו מחלקה PetMaster והפעם תבחרו אופציה ב-Eclipse שמייצרת תכונה main(). תשנו קוד שנוצר בעזרת Eclipse שיראה כך:

```
} Public class PetMaster
} (Public static void main(Strin[] args
;String petReaction
;()Pet myPet = new Pet
;()MyPet.eat
;("!!PetReaction = myPet.say("Tweet!! Tweet
;(System.out.println(petReation

;()MyPet.sleep
{
{
```

לא לשכוח להקיש Ctrl-S לשמירה וקומפילציה של המחלקה הזאת! להרצת מחלקת PetMaster יש לבחור בתפריט של Run,Run...,New Eclipse ולהקיש שם של המחלקה ראשית: PetMaster.
ללחוץ על הכפתור Run והתוכנית תדפיס את הטקסט הבא:
אני כל כך רעב... תן לי לאכול חטיף!
טוב!טוב! צוויץ!! צוויץ!!
לילה טוב, אראה אותך מחר

PetMaster היא מחלקה קוראת ו מתחילה עם יצירת instance של אובייקט Pet . היא מגדירה משתנה MyPet ומשתמשת בפקודה של new Java:

```
;()Pet myPet = new Pet
```

השורה הזאת מגדירה משתנה מסוג Pet (כן,זה נכון, אפשר להתייחס לכל המחלקות שנוצרו כמו סוג נתונים חדש של Java). עכשיו המשתנה myPet יודע איפה בזיכרון של מחשב נוצר instance של Pet וכך אפשר להשתמש במשתנה הזה כדי לקרוא לשיטה של מחלקה Pet, למשל:

```
;()MyPet.eat
```

אם שיטה מחזירה ערך, צריך לקרוא לשיטה הזאת בצורה אחרת. צריך להגדיר משתנה מאותו סוג של ערך מוחזר משיטה ולייחס אותו למשתנה הזה. עכשיו לשיטה אפשר לקרוא בצורה כזאת:

```
;String petReaction
)petReaction = myPet.say "צוויץ!!צוויץ!!");
```

בשלב זה הערך המוחזר נשמר במשתנה petReaction ואם אתם רוצים לראות מה יש פה, בבקשה:

```
;(System.out.println(petReaction
```

ירושה – דג הוא גם חיית מחמד

המחלקה שלנו Pet תעזור לנו ללמוד עוד תכונה חשובה של Java, שנקראת ירושה (inheritance). בעולם האמיתי, כל אחד יורש איזה שהם תכונות מההורים שלו. בעולם של Java אתם גם יכולים ליצור מחלקה חדשה שמבוססת על אחת שקיימת.

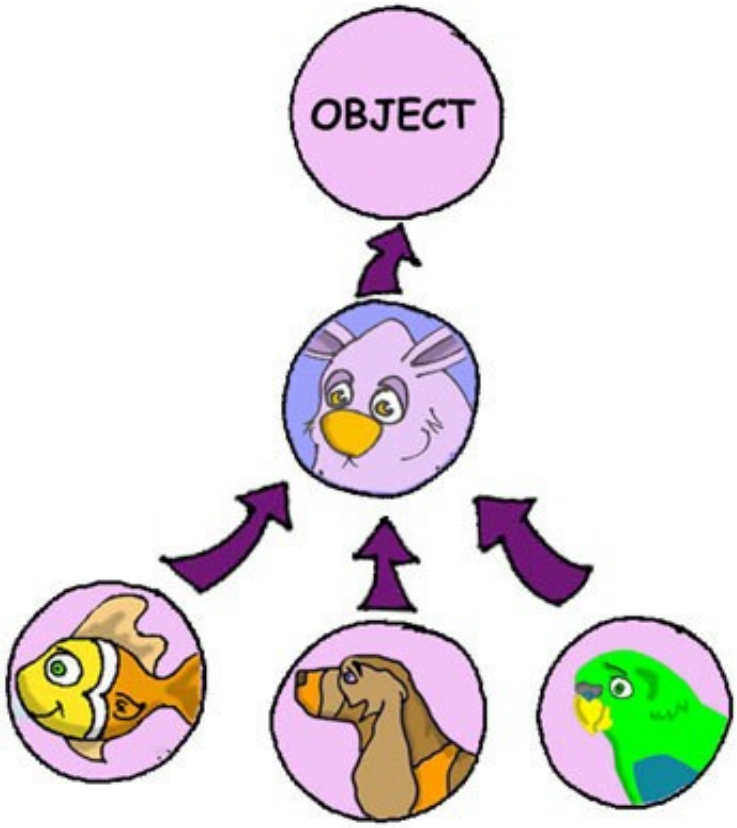
למחלקה Pet יש התנהגות ומאפיינים שקיימים אצל הרבה חיית מחמד – הם אוכלים, ישנים, יש כאלה שמשמיעים קולות, לעור שלהם יש צבע שונה וכו'. מצד שני החיית הם שונות – כלבים נובחים, דגים שוחים ולא משמיעים קולות, תוכים מדברים יותר טוב מכלבים. אבל הם כולם אוכלים, ישנים, יש להם משקל וגובה. לכן יותר קל ליצור מחלקה שתירש התנהגויות ומאפיינים ממחלקה Pet במקום ליצור Dog, Parrot או Fish מחדש כל פעם.
מלת המפתח extends תעשה את זה:

```
} class Fish extends Pet
  {
```

אתם יכולים להגיד ש Fish- שלנו, היא תת מחלקה של מחלקה Pet, ומחלקת Pet היא סופרקלס של מחלקת Fish. במילים אחרות מחלקה Pet היא כמו תבנית לייצור מחלקה Fish . אם אתם תשאירו מחלקה Fish כמו שהיא, אתם עדיין יכולים להשתמש בכל התכונות והשיטות שעוברים בירושה ממחלקה Pet .
תסתכלו:

```
;Fish myLittleFish = new Fish
;()MyLittleFish.sleep
```

אפילו שאנו לא הגדרנו אף שיטה במחלקת Fish , אנו יכולים לקרוא לשיטה sleep() מהסופרקלס שלה!
יצירת תת מחלקה ב-Eclipse זה פשוט! בחר בתפריט File,New,Class ותקיש Fish , בתור שם של מחלקה. תשנה Java.lang.Object בשדה סופרקלס למילה Pet.



על תשכח שאנו מיצרים סופרקלס Pet כדי להוסיף תכונות חדשות שקיימות רק אצל דג ולהשתמש בקטעי קוד שכתבנו לחית מחמד כללית.

זה הזמן לגלות סוד – כל מהחלקות ב-Java עוברים בירושה מסופרקלס Object, לא משנה אם השתמשו במילה Extention או לא.

אבל למחלקות Java לא יכולות להיות שני הורים. אילו זה היה קורא אצל בני אדם, ילדים לא יכלו להיות תת מחלקות של ההורים שלהם, אבל כל הבנים היו צאצאים של אדם וכל הבנות היו צאצאים של חווה.

לא כל החיות יכולות לצלול אבל דג כן יכול. בו נוסיף שיטה חדשה dive() למחלקה Fish.

```
}Public class Fish extends Pet
;Int currentDepth=0
} (Public int dive(int howDeep
```

```
;CurrentDepth=currentDepth + howDeep
;(System.out.println(Diving for  + howDeep + feet
;(System.out.println(Im at + currentDepth + feet below sea level
;Return currentDepth
{
  {
```

לשיטה dive() יש ארגומנט howDeep שאומר לדג עד כמה עמוק הוא יכול לצלול.
אנו גם מגדירים משתנה של מחלקה currentDeep שתשמור ותעדכן עומק נוכחי כל פעם שאנו קוראים לשיטה dive(). השיטה הזאת מחזירה ערך נוכחי של משתנה currentDepth למחלקה הקוראת.

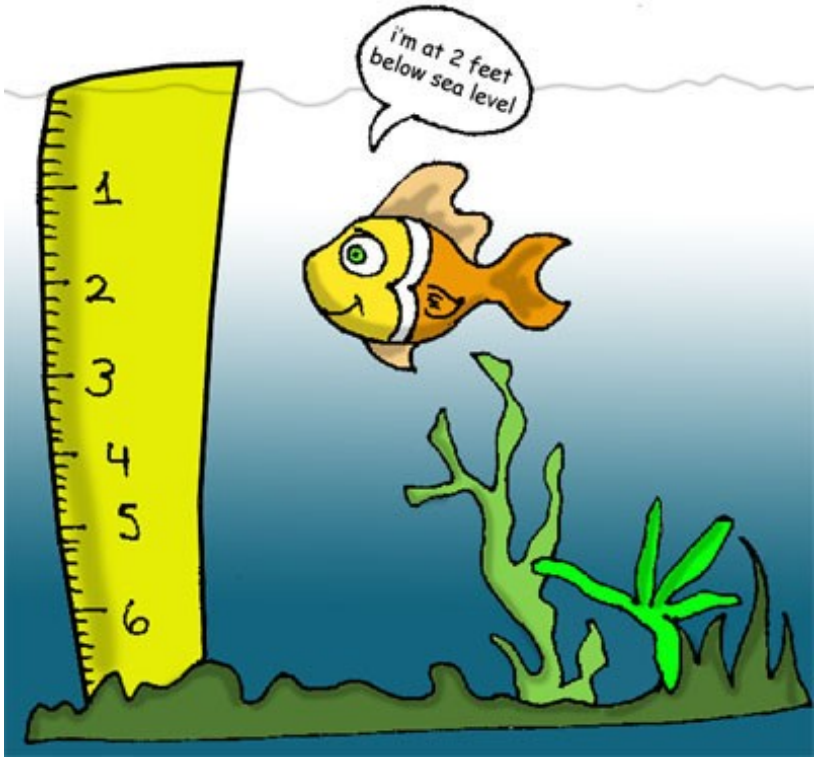
נא ליצור מחלקה אחרת FishMaster שתיראה כך:

```
} Public class FishMaster
}   (Public static void main (String[] args
;()Fish myfish = new Fish
;(MyFish.dive(2
;(MyFishdive(3
;()MyFish.sleep
{
 {
```

השיטה main() עושה instance של אובייקט Fish וקוראת לשיטה שהאובייקט פעמיים, עם ארגומנטים שונים. אחרי זה היא קוראת לשיטה sleep() . כאשר תריצו תוכנית FishMaster, היא תדפיס את המשפטים הבאים:

```
Diving for 2 feet
Im at 2 feet below sea level
Diving for 3 feet
Im at 5 feet below sea level
Good night, see you tomorrow
```

האם שמתם לב שחוץ משיטות מוגדרות במחלקה Fish , FishMaster גם קוראת לשיטות מסופרקלס Pet? כל זה העניין של הירושה – אין צורך להעתיק את הקוד ממחלקת Pet , רק להשתמש במילה Extends, ומחלקת Fish יכולה להשתמש בשיטות של Pet!



דבר נוסף, אף על פי שהשיטה dive() מחזירה ערך של currentDepth, Fishmaster שלנו לא משתמש בו. זה בסדר, Fishmaster שלנו לא צריך את הערך הזה, אך יכול להיות שמחלקות אחרות ישתמשו ב-Fish וירצו להשתמש בערך מוחזר. למשל, תחשבו על מחלקה FishTrafficDispatcher שצריכה לדעת את המצב של דג לפני שתאפשר צלילה של דג אחר בשביל למנוע תאונה.

שיטת דריסה

כמו שידוע לכם, דג לא מדבר, (לפחות לא בקול רם...). אבל המחלקה שלנו Fish עברה בירושה ממחלקת Pet ובה יש שיטה say(). לכן שום דבר לא יכול למנוע מכם מלכתוב משהו כמו זה:
;()MyFish.say

וכעת, הדג שלנו התחיל לדבר... אם אתם לא רוצים שזה יקרה צריך שמחלקת Fish תדחה את שיטת Pet say(). זה עובד כך: אם מגדירים בתת מחלקה שיטה עם אותה חתימה כמו בספרקלס, השיטה של תת מחלקה תפעל במקום שיטה של סופרקלס. בו נוסיף שיטה say() למחלקה Fish.

```
}(Public String say(string something
;"?Return "Don't you know that fish do not talk
{
```

עכשיו תוסיפו קריאה לשיטה main() הבא משיטה של מחלקה FishMaster :
;(MyFish.say(Hello

תריצו תוכנה והיא תדפיס:

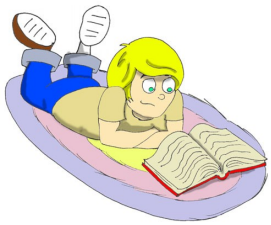
?Don't you know that fish do not talk

זה מוכיח ששיטה say() של Pet נדרסה .

אם חתימת השיטה כוללת מילת מפתח final , שיטה כזאת אי אפשר לדרוס למשל:
{...} ()Final public void sleep

וואו

למדנו הרבה בפרק הזאת – בא ניקח הפסקה.!

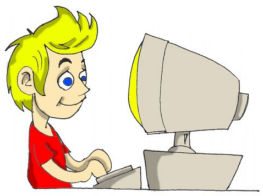


קריאה מומלצת

א.סוגי משתנים של JAVA

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html

ב.על ירושה
http://java.sun.com/docs/books/tutorial/java/concepts/inheritance.html



תרגול

א. צרו מחלקה חדשה בשם Car: עם השיטות הבאות

```
public void start()
public void stop()
public int drive(int howlong
```

שיטה drive() צריכה להחזיר את המרחק כולל שעברה המכונית, בזמן נתון.
תשתמשו בנוסחה הבאה של חישוב מרחק:

```
distance = howlong*60;
```

ב. כתבו עוד מחלקה carowner שמייצרת מופע של אובייקט Car ותקראו את השיטות שלו.
להדפסת תוצאת כל השיטה תשתמשו ב- System.out.println() .

בונוס ל"חוכמולוגים"

צרו תת-מחלקה של Car שנקראת JamesBondCar ותדרסו את שיטת drive() .
תשתמשו בנוסחה הבאה לחישוב מרחק:

distance = howlong*180;

תהיו יצירתיים, תדפיסו מסרים מצחיקים!