**Java Class Structure(מבנה המחלקה)**

Field and Methods in Java (שדות ומתודות)

כל דבר בג'אווה קשור למחלקות ואובייקטים, לכל מחלקה יש תכונות ושיטות.

לדוגמא: בחיים האמיתיים, מכונית היא אובייקט. למכונית יש תכונות, כמו משקל וצבע, ושיטות, כמו נסיעה ובלימה.

מחלקה היא כמו בונה אובייקטים, או "שרטוט" ליצירת אובייקטים.

אם לדוגמא נרצה להגדיר מחלקה חדשה של מכונית זה ייראה כך:

class Car {

}

לכל מחלקה ניתן להגדיר כמות בלתי מוגבלת של תכונות עם סוג ואותם נגדיר בתחומים שנקראים "שדות"

class Car {

// Field שדה

String weight;

String color;

{

סוגים פרימיטיביים וסוגי "אזכור"

בגאווה יהיה לנו סוגים של משתנים הנקראים "פרימיטיביים" להם שמות שמורים לדוגמא

Int, char, long בהם נשתמש כדי להגדיר למשל מספרים ואותיות.

סוגי "אזכור" הם סוגי משתנים שמצביעים לכתובת בזיכרון השמורה לאובייקט מסוים.

לדוגמא String, אובייקט זה הוא "אוסף" של משתנים פרימיטיביים מסוג char המשורשרים יחדיו למילה, כך שנוכל להשתמש בString בשביל תכונות מילוליות ולא בהמון אותיות לחוד.

class Car {

// Field שדה

String color;

Int mileage;

}

ניתן לראות כי סוגים פרימיטיביים כמו int יוגדרו עם אותיות קטנות, וסוגי אזכור כמו String ומחלקות שונות מיניהם יוגדרו עם אות גדולה בהתחלה.

חוץ מתכונות המחלקה, יהיו לה גם את השיטות שלה , שיטה היא מתודה, המוציאה לפועל משימות. לדוגמא הרכב שלנו נוסע, אז נגדיר לו שיטה חדשה בצורה הבאה:

class Car {

// Field שדה

String color;

Int mileage;

// Methods שיטות

void drive() {

}

}

כפי שניתן לראות האזור בו הגדרנו את התכונות, ייקרא שדה והאזור בו נגדיר את השיטות ייקרא אזור המתודות.

שיטה רק תוציא לפועל את הפעולות שבתוכה אם נגדיר אותה כ void .

לשיטה יהיה אפשר לתת סוג אחר ולא void , במקרה זה נצטרך גם להחזיר בתוך השיטה אובייקט מאותו סוג עם המילה השמורה return

String drive() {

return ”driving”;

}

שיטה זו שווה לתכונה הבאה, שתיהן מחזירות את אותה המילה:

String drive = “driving”;

ההבדל בין מחלקה בגאווה לקובץ בגאווה

קובץ בגאווה יהיה מוגדר עם הסיומת .java ואם נרצה להגדיר מחלקה בתוך הקובץ, נהיה מחויבים ליצור מחלקה עם שם הקובץ.

חוץ מהמחלקה שנהיה חייבים להגדיר עם שם הקובץ, נוכל באותו הקובץ להגדיר מספר מחלקות.

בנאים

תכונות מחלקת הרכב שכתבתנו מוגדרות למחלקה אך אין להם ערכים, כלומר הערכים שלהם לא מוגדרים. לשם כך נועדו הבנאים של המחלקות, בנאי יוגדר באופן דומה לשיטה, אך בלי סוג ושם השיטה יהיה זהה לשם המחלקה, אם לא נגדיר בנאי בעצמינו, גאווה לבד תיצור בנאי ריק בעת ההרצה של הקוד (אנחנו לא נראה את הבנאי הזה) ואם נגדיר אותו זה יהיה אותו הדבר כמו :

class Car {

// Field שדה

String color;

Int mileage;

// Constructor בנאי

Car() {

}

// Methods שיטות

void drive() {

}

}

במקום לתת למחשב ליצור לנו בנאי ריק, אנחנו ניצור בנאי עם פרמטרים הזהים לתכונות המחלקה

class Car {

// Field שדה

String color;

Int mileage;

// Constructor בנאי

Car(String color,int mileage) {

this.color = color;

this.mileage = mileage;

}

// Methods שיטות

void drive() {

}

}

הוספנו לבנאי 2 פרמטרים, ועם המילה השמורה this הגדרנו את תכונות המחלקה לפני אותם פרמטרים.

The main method

בכל פרויקט יהיה לנו את השיטה main שתרוץ ראשונה ותדאג להוציא לפועל את שאר הקוד,

המתודה תוגדר באיזה קובץ שנבחר בצורה הבאה:

public static void main(String[] args) {

}

בתוך מתודה זו נוכל להתחיל להוציא לפועל כל מני פעולות וליצור אובייקטים חדשים, לשם הדוגמא אנחנו נרצה להדפיס לקונסולה, נשתמש בשורה הבאה כדי להדפיס שורות:

System.out.println(“Hello World”);

נוסיף שורה זו לשיטהmain וכשהקוד ירוץ נקבל את ההדפסה בקונסולה “Hello World”

public static void main(String[] args) {

System.out.println(“Hello World”);

}

כמו שהדפסנו Hello World בשיטה main נוכל גם ליצור באותה שיטה "מופע" חדש של המחלקה Car : אם לא סיפקנו בנאי במחלקה כך ייראה מופע חדש של המחלקה:

public static void main(String[] args) {

Car car = new Car();

}

אם סיפקנו בנאי שמקבל פרמטרים נצטרך להעביר אותם בעת יצירת מופע חדש:

public static void main(String[] args) {

Car car = new Car(“Red”,1000);

}

השתמשנו בבנאי שמקבל 2 פרמטרים, במקרה שלנו String, int המציינים את צבע הרכב ואת הקילומטראז.

כעת יהיה אפשר לגשת לתכונות המחלקה ולשיטותיה:

נרצה להדפיס את צבע הרכב:

public static void main(String[] args) {

Car car = new Car(“Red”,1000);

System.out.println(car.color);

}

בעת הרצה נקבל את ההדפסה בקונסולה "”Red