ממשק interface

ממשק הוא כלי שמסייע לנו כאשר אנו מעוניינים שאובייקט כלשהו יקיים פעילות או אוסף מסוים של פעילויות.

למשל: אם אנו רוצים שאובייקט יהיה בר הדפסה. נכתוב ממשק שמכיל הכרזה על פונקציה ריקה המדפיסה את תוכן האובייקט. כעת, אנו יודעים כי כל אובייקט שנוצר ממחלקה המממשת את הממשק שלנו, מכיל פונקציה המאפשרת את הדפסתו (כמובן, כל אובייקט מממש את השיטה הזו בדרכו - אי אפשר לדעת באיזו דרך מומשה הפונקציה).

ממשק יכול להכיל **פונקציות אבסטרקטיות בלבד**, והוא מגדיר את המונח "**מה המחלקה צריכה לעשות**". כל מחלקה המממשת את אותו הממשק, מתחייבת לממש את כל התכונות או הפונקציות המוגדרות בו. (ממשקים מתארים פונקציונליות ויישום הממשק על ידי מחלקה מעיד כי יש לה את אותה התנהגות פונקציונלית מסוימת).

בכדי להבין את העניין, ניקח לדוגמה את קבוצת הפעולות הבאות:

- moveLeft
- moveRight
- moveForward
- moveBackward

האובייקטים שיכולים לממש את קבוצת הפעולות האלו הם:

- אדם - וכל הנגזרות של האדם
- חיה- וכל הנגזרות של החיה
- כלי רכב - וכל הנגזרות של כלי הרכב
- חייל במשחק שחמט
- שואב אבק רובוטי
- ועוד...

כאשר נתבונן ברשימת האובייקטים הנ"ל, לא נמצא קטגוריה מהותית משותפת לכל אותם אובייקטים, מלבד היכולת המשותפת שלהם לנוע ימינה שמאלה אחורה וקדימה.

(תזכורת: כלל ההורשה מתקיים כאשר הנגזרת היא גם הבסיס, ובמקרה זה לא מתאים ליצור מחלקת בסיס שתוכל להתאים גם לחיה, גם לשואב אבק וכו').

בצורת עבודה ללא ממשקים - היינו פונים לכל מחלקה, ומוסיפים לה את הפונקציות בנפרד.

אבל, צורת כתיבה זו יכולה לצאת מכלל שליטה כאשר נטפל בקוד מורכב, המכיל מחלקות רבות ואין ספור התנהגויות למימוש.

בנקודה זו יש לנו הזדמנות להשתמש בממשק - כאשר זיהינו דפוס התנהגות משותף (בדוגמה שלנו זו הפונקציונליות של התנועה הבסיסית), אנו יכולים ליצור ממשק בשם Imove, שיכיל את כל הפעולות והיכולות המשותפות (לנוע קדימה אחורה שמאלה וימינה), וכל מחלקה שתממש את הממשק Imove בהכרח תהיה חייבת לממש את כל הפונקציות של אותו ממשק, בהתאם למשמעות שיש לפעולה באותה מחלקה.

הערה: כאשר אנו רוצים לאפשר גמישות בקוד שאנו כותבים, ובנוסף להגדיר סטנדרטיזציה עבור מתכנתים אחרים שעובדים עם הקוד שלנו. נוכל באמצעות ממשקים להגדיר תקן מימוש וספציפיקציות של שמות פונקציות והפרמטרים שלהם.
צורת עבודה זו מבטיחה באמצעות הקומפיילר כי כל מחלקה המממשת את הממשק יישמה את כל הפונקציות המוגדרות בו. ולכן נהוג לכנות את הממשק בכינוי contract שפירושו חוזה (כל מחלקה המממשת את הממשק מתחייבת לחוזה שהיא תממש את כל הפונקציות המוגדרות בממשק)

ההבדלים בין Abstract לבין Interface :

	Abstract class	Interface
מהות השימוש	מחלקה אבסטרקטית מגדירה את הזהות הבסיסית של הנגזרות, ומשמשת מחלקת בסיס לאובייקטים מאותו סוג (טיפוס). ומכיוון שמחלקה אבסטרקטית מגדירה את הזהות הבסיסית של הנגזרת - מהמחלקה האבסטרקטית Person (שמכילה את המאפיינים גיל ומשקל), לא נכון ליצור נגזרת Cat (אפילו שגם היא מכילה גיל ומשקל)	ממשקים משמשים להגדרת היכולות ההיקפיות של המחלקה. במילים אחרות, ממשקים מתארים פונקציונליות- מה אובייקט יכול לעשות ולא מהו ולכן, הן אדם והן רכב יכולים לרשת מממשק IMovable
הורשה מרובה (Multiple inheritance)	מחלקה יכולה לרשת רק מחלקה אבסטרקטית אחת	מחלקה יכולה לממש כמה ממשקים (כאשר בין ממשק לממשק יפריד פסיק)

ירושה מממשק	מימוש ממשק ע"י מחלקה אבסטרקטית נעשה ע"י המילה implements
	`interface IA { }`
	`interface IB { }`
	`interface IC { }`
	`abstract class A implements IA{ }`
	`abstract class B extends A implements IB, IC{ }`
		ירושת ממשק ע"י מממשק נעשה ע"י המילה extends
		`interface IA { }`
		`interface IB { }`
		`interface IC extends IA , IB{ }`

מימוש פונקציות מופשטות בנגזרות	חובה לממש את כל הפונקציות האבסטרקטיות של המחלקה האבסטרקטית (מחלקה אבסטרקטית לא חייבת לממש פונקציות אבסטרקטיות של המחלקה האבסטרקטית שלה, והן יהיו חייבות להיות ממומשות בנגזרות)	חובה לממש את כל הפונקציות האבסטרקטיות של הממשק (מחלקה אבסטרקטית לא חייבת לממש פונקציות אבסטרקטיות של הממשק – והן יהיו חייבות להיות ממומשות בנגזרות)
מימוש דיפולטיבי לפונקציה	מחלקה אבסטרקטית יכולה להכיל פונקציות ממומשות או פונקציות אבסטרקטיות	ממשק יכול להגדיר רק פונקציות אבסטרקטיות
הגדרת הרשאות גישה	פונקציה אבסטרקטית לא תהיה private, כל שאר הפונקציות והמאפיינים יכולים להיות מוגדרים בכל הגדרת גישה שנרצה	בממשק אין אפשרות להגדיר הרשאת גישה private, (הכל יהיה אוטומטית public)
תכונות ופונקציות סטטיות	במחלקה אבסטרקטית אפשר ליצור מאפיינים סטטיים וכן פונקציות סטטיות. (רק פונקציה אבסטרקטית לא יכולה להיות סטטית)	ממשק לא יכול להכיל מאפיינים סטטיים / פונקציות סטטיות
בנאי	יכולה להכיל בנאי. (הבנאי יבוצע ע"י הנגזרת)	ממשק לא יכול להכיל בנאי
יצירת מופעים	אין ליצור מופעים	אין ליצור מופעים

דוגמה מלאה למימוש ממשקים ומחלקות:

- ניצור ממשק בשם IWash המכיל את הפעולות שטיפה וייבוש.

- ניצור מחלקה אבסטרקטית בשם Person, ומכיוון שכל האנשים בעולם מממשים את סדרת הפעולות של הממשק IWash, נגדיר ישירות שPerson מממש את IWash.

  למחלקה אבסטרקטית זו ניצור את הנגזרת Child.

- ניצור מחלקה אבסטרקטית בשם HomeProduct, ולא נגדיר ישירות שHomeProduct מממש את IWash. מכיוון שלא כל חפצי הבית מממשים את סדרת הפעולות של הממשק IWash (לדוגמה: לא נשטוף ספרים)

  למחלקה אבסטרקטית זו ניצור את הנגזרות Cupו- Coat.

להלן הדיאגראמה של היררכית ההורשה בתכנית: