# חלק יבש - *Types and values*

## Kotlin

1. **קראו על בקרת זרימה (Control Flow) בשפת Kotlin.**

**מהו הדמיון ומהו השוני בין if-else של Kotlin לבין if-else של שפת C?**

**הדומה** - המבנה זהה, כל if-else בשפת C ניתן לרשום גם ב-kotlin.

**השונה** - If-else בשפת Kotlin הוא מיוחד משום שאנחנו יכולים להתייחס ל-if-else כביטוי. לדוגמא, if-else מחזיר ערך אשר אנחנו יכולים לתת לערך או משתנה, לכן אנחנו לא צריכים אופרטור טרינרי ב-kotlin. בשונה משפת C.

**מהו הדמיון ומהו השוני ביחס ל if-then-else של ML?**

**הדומה** – בשתי השפות,

**השונה** - ב-ML ה- If-elseדומה לטרינרי, מכיוון שהוא תמיד מחזיר ערך (שפה פונקציונאלית).

**האם לאופרטור הטרינארי בשפת C קיים מקביל בשפת Kotlin ? אם כן מהו ? אם לא, מדוע?**

האופרטור הטרינארי – אופרטור אלביס, קיים בkotlin:

האופרטור אלביס מחזיר את הצד השמאלי אם הוא לא NULL, ואת הצד הימני אחרת: בדר"כ משתמשים בו עם return:

Val foo = bar() ?: return

1. **האם פונקציה היא טיפוס בשפה? הסבירו.**

התשובה היא כן, פונקציה בkotlin מוגדרת ע"י סוגי הטיפוסים אותם היא מקבלת, והטיפוס אותו היא מחזירה. אזי, פונקציה יכולה להשתייך לקבוצה, כלומר, בעלת טיפוס.

**האם פונקציה ב Kotlin יכולה להיות פולימורפית ? אם כן, מה ההבדל בין פונקציה שכזו ב Kotlin לבין פונקציה פולימורפית בשפת ML?**

התשובה היא כן, פונקציה בkotlin יכולה להיות פולימורפית, למשל פונקציית sort לArray.

בשפת Kotlin נידרש להכריז לפני קריאה לפונקציה פולימורפית על סוג המשתנה ״T״ עליו הפונקציה תתבצע, לעומת ML שם אין צורך להכריז כל פעם על סוג המשתנה, אלה המפרש מבצע זאת באופן עצמאי(בתנאי שהמשתנה המתאים לדרישות הפונקציה, למשל אם בפונקציה יש שימוש באופרטור השוואה אז למשתנה שמוכנס חייב להיות אופרטור השוואה).

**ב Kotlin ניתן להגדיר פונקציה ב infix notation, השוו בין הגדרה שכזו בשפת ML לבין זו שב Kotlin (ההשוואה צריכה לכלול : מתי ניתן להגדיר, ואיך ההגדרה מתבצעת)**

* Kotlin: ההגדרה על Infix מתבצע בהכרזה על הפונקציה הרלוונטית, בנוסף חייב לשייך את הפונקציה לטיפוס מסוים(טיפוס הthis), הגדרה: בתחילת הפונקציה ע״י המילה השמורה infix.
* ML: ההגדרה על Infix יכול להיות מוגדר בכל מקום, כלומר גם לפני ההגדרה על הפונקציה וגם אחרי. הגדרה: ע״י שימוש במילה השמורה infix ואחריה השם של הפונקציה/תו הפעולה(למשל +).

1. **הסתכלו בשקפים של פרק 3. מהי תכונת Void Safety?**

בטיחות void יכולה להיות בכל שפה, גם אם היא לא מונחית עצמים. בדרך כלל, מדובר על שפות שיש בהן reference semantics.. בשפות כאלו, יכולה להיות בעיה, והבעיה אכן קורית, של גישה למצביע שהוא NULL

המונח VOID SAFETY מתייחס למאמצים למנוע שגיאה של גישה לערך שהוא NULL עוד בזמן הידור, תוך שימוש בבדיקת טיפוסים סטטית.

**האם Kotlin היא Void Safe? אם כן, הסבירו כיצד תכונה זו באה לידי ביטוי בשפה. אם לא, תנו דוגמת קוד שמוכיחה זאת. אל תפספסו את האופרטור של Elvis.**

ברוב המקרים Kotlin היא Void/Null safe, מכיוון שהשפה מוכוונת להיות Null safe לעומת Java כדי

לא לבצע בדיקות exception מיותרות. תכונה זו באה לידי ביטוי ע״י שימוש בסימן “?” לפי משתנים,

למשל: b?.length כאשר b הוא מחרוזת. במידה וb לא מאותחל, לא יהיה כניסה לlength.

אך למרות הניסיון להפוך את השפה לVoid Safe, אפשר לעקוף זאת ע״י שימוש בסימן ״!!״ אחרי משנה,

או כאשר יש שימוש במודולים חיצוניים שנכתבו בשפת Java.

1. **קראו את** [**הסיכום על**](https://docs.google.com/document/d/11kTXneoHLijenMyMzl1ED6ih8tpb2zrDvE78mw8N4wM/edit) **[פולימורפיזם](https://docs.google.com/document/d/11kTXneoHLijenMyMzl1ED6ih8tpb2zrDvE78mw8N4wM/edit). מהם סוגי הפולימורפיזם הקיימים בשפת Kotlin, הסבירו בפירוט והביאו דוגמאות לכל אחד מן הסוגים (לפחות דוגמה אחת לכל סוג)**

* Overloading:

פעולת העמסה על פעולות(operators) מתבצע ע״י הגדרת פונקציה על האופרטור עם טיפוסים אחרים/חדשים. למשל: a.plus(b) יבצע העמסת אופרטור על ״+״.

* Coercion:

בKotlin אין המרת טיפוסים אוטומטית, לכן פונקציה המקבלת Int לא תקבל double במקום.

עם זאת, קיים בkotlin המרות בין טיפוסים ע״י פונקציות מתאימות,

למשל: המרה של int לchar מתבצע ע״י Int.toChar()

* Parametric:

בKotlin קיימת אפשרות להגדרת פונקציות המקבלות/מחזירות משתנה T, כלומר פונקציה העובדת על מספר רב(לא מוגבל) של טיפוסים.

למשל: פונקציית sort למערכים, fun <T> Array<out T>.sort() יכולה לבצע מיון של מערך למספר לא מוגבל של מערכים(עם טיפוסים שונים).

* Subtyping:

שפת Kotlin מבוססת OPP, וכל מערכת הטיפוסים שלה מסוג אובייקטים, למשל: Int יורש מNumber שיורש מAny.

לכן הSubtyping מובנה בשפה, ולמשל: מערך של Any יכול להחזיר איברים מטיפוס Int.