5. לקראת תחרות ריצת מרתון הוגדרה מחלקה **Competitor** המייצגת מתחרה שסיים את המסלול.

למחלקה יש שלוש תכונות:

- minutes — מספר הדקות שנדרשו למתחרה לסיים את המסלול (מספר הדקות אינו מוגבל ל־ 60), מטיפוס שלם.
- seconds — מספר השניות שנדרשו למתחרה לסיים את המסלול (מספר השניות הוא עד 59 כולל), מטיפוס שלם.
- name — שם המתחרה מטיפוס מחרוזת.

הנח שלכל תכונה הוגדרו בשפת Java הפעולות get ו־ set ובשפת C# הפעולות Get ו־ Set .

נוסף על כך הוגדרה מחלקה בשם **Race** המאגדת אוסף של כל המתחרים שסיימו את המסלול. מספר המסיימים אינו ידוע.

הנח שאין שני מתחרים שסיימו את המסלול בזמן זהה.

לפניך ממשק חלקי של המחלקה **Race** הכתוב בשפת Java ובשפת C# :

סיבוכיות	תיאור הפעולה	כותרת הפעולה
O(n)	הפעולה מקבלת עצם מטיפוס **Competitor** ומוסיפה אותו לאוסף. שים לב: הפעולה הבאה של הממשק (rank בשפת Java או Rank בשפת C#) ודרישות הסיבוכיות שלה רלוונטיות לאופן שבו מממשים פעולה זו.	בשפת **Java** — public void add (Competitor x) בשפת **C#** — public void Add (Competitor x)
O(n)	הפעולה מקבלת דירוג ומחזירה את שם המתחרה, שלו דירוג זה באוסף. הדרכה: 1 הוא המתחרה שסיים את המסלול בזמן הקצר ביותר, 2 הוא המתחרה שסיים את המסלול בזמן השני הקצר ביותר וכן הלאה. הנח שקיים מתחרה בדירוג המבוקש. הערה: אין למחוק איברים מהאוסף.	בשפת **Java** — public String rank (int x) בשפת **C#** — public string Rank (int x)

שים לב:

- לכל פעולה יש הנחיות סיבוכיות זמן ריצה.
- n הוא מספר המתחרים באוסף.
- **חובה לעמוד בדרישות הסיבוכיות**.

בעבור הסעיפים א-ב עליך להשתמש במבנה נתונים מתאים העומד בדרישות השאלה.

תוכל להשתמש בכל מבנה נתונים שלמדת.

אפשר לכתוב פעולות במחלקה **Competitor** .

א. כתוב את תכונות המחלקה **Race** .

ב. מַמֵּש את פעולות המחלקה **Race** המופיעות בממשק המחלקה שבשאלה.