**עבודה 2 תכנות מערכות**

1 א.

התופעה היא העברת זמן מעבד על ידי ה-scheduler, והיא קוראת כאשר הזמן שהמעבד מקצה עבור כל תהליכון עובר מתהליכון אחד לאחר על ידי המתזמן של מערכת ההפעלה. התופעה עשויה לקרות בכל זמן שהתהליכונים רצים.

1ב.

התופעה כן עלולה ''להחמיר''. אם שני התהליכונים שותפים לאותו זיכרון וקטע הקוד שמעדכן את מיקומי המשתתפים שסיימו לא מסונכרן, הדבר עלול לגרום לכך שמשתתף 4 עשוי להגיע ל-100 מטרים ובשינוי זמן המעבד עבורו, לא יכנס לקטע הקוד שגורם לו לסיים את המרוץ. לעומת זאת משתתף 3 ,שעשוי לקבל את זמן המעבד, עלול להגיע ל-100 מטרים ובנוסף להיכנס לקטע הקוד שמעדכן את מסיימי המרוץ ובכך ולקבל את מקומו של משתתף 4.

2.

לא. בגלל שסיום מרוץ מהר יותר תלוי במספר גדול יותר של גורמים הקובעים את קצב התקדמות התהליכון. לדוגמא- סיבוכיות התוכנית: משך הזמן שלוקח לתהליכון לבצע פעולה מסוימת או בסדר החלוקה של זמן המעבד שקובע המתזמן של מערכת ההפעלה.

3א.

שיטה ראשונה -ייבוא מחלקת Thread:

ייבוא המחלקה Thread מאפשרת תפעול גישה מהירה לתהליכון ולבצע בו מניפולציות שונות ישירות מהמחלקה. חיסרון של גישה זו שהיא אינה מאפשרת ייבוא של מחלקות נוספות.

שיטה שנייה - יישום הממשק Runnable :

יישום הממשק Runnable מאפשרת גמישות של המחלקה בכך שהוא מאפשר ליישם מגוון של ממשקים בנוסף לממשק, בנוסף הוא גם מבודד את התנהגות התהליכון מכל המחלקה ובכך מאפשר סדר וכמוס.

חיסרון של הממשק Runnable הוא שאין שליטה על התהליכון בכך שמתודות כמו sleep() או join() לא נמצאות בממשק.

3ב.

כאשר ניצור מופע ממחלקה המיישמת Runnable נחויב גם ביצירת מופע של המחלקה Thread בגלל שאין לממשק Runnable את האפשרות להתחיל ולנהל את ביצוע הרצת התהליכון בעוד שלמחלקה Threat יש. כדי שנוכל לשלוט בתהליכון נצטרך אובייקט מהמחלקה.Thread