שאלה 1

א. תחילה חשבנו להוסיף את הפונקציונאליות הטריוויאלית עבור גרף באשר הוא – הוספה והסרה של מה שמרכיב גרף, צמתים וקשתות. בנוסף, בהתאם לדרישה, הוספנו את האפשרות לקבל בנים או אבות של צומת מסוימת, או לקבל בן או אב של צומת התואם לקשת שיוצאת או נכנסת אליה. בהתאם לדרישה שכל צומת בגרף יכולה לייצג כל אובייקט שהוא על מנת למדל מערכות בעזרת גרף דו צדדי, הוספנו אפשרות לקבל את האובייקט המיוצג על ידי הצומת או לשנות את אובייקט זה. כמו כן, מתודות שמבצעות פעולות על הגרף מחזירות חיווי על הצלחת השינוי, לדוגמא נקבל ערך בוליאני שמעיד על האם הכנסת צומת חדשה הצליחה.

ב. ראשית, הגדרנו מחלקה בשם Node שמממשת צומת בגרף. מחלקה זו מממשת את הפעולות הבאות:

```
Node(K label, Object data, boolean isBlack)
boolean isBlack()
Object getData()
void setData(Object data)
K getLabel()
void setLabel(K label)
Map<K, Node<K>> getChildrenEdges()
List<K> getChildrenLabels()
void addChild(K edgeLabel, Node<K> child)
Map<K, Node<K>> getParentsEdges()
List<K> getParentsLabels()
void addParent(K edgeLabel, Node<K> parent)
void removeChild(K label)
void removeParent(K label)
```

מחלקה זו היא הפשטה של צומת והיא שומרת בתוכה עבור על צומת את התווית שלו, המידע שהוא מחזיק, מפות שמפתחותיהן הם תוויות של קשתות וערכיהן הם צמתים אחרים (אחת לקשתות יוצאות ואחת לקשתות נכנסות), רשימות של תוויות הצמתים הבנים והאבות ומשתנה בוליאני המגדיר האם הצומת הוא שחור או לא.
בחירה במימוש זה מקלה על שליפת הרשימות בסיבוכיות זמן קבועה, שכן הם נשמרים ברשימות והרשימות מוחזרות מיד. המפות מאפשרות גישה מהירה וניהול פשוט של קשרי משפחה בין הצמתים השונים.

לאחר מכן, ניגשנו למימוש BipartiteGraph. המחלקה מממשת את הפעולות הבאות:

```
BipartiteGraph()
Node<L> findNode(L label)
boolean addNode(L label, Object data, boolean isBlack)
boolean removeNode(L label)
boolean addEdge(L from, L to, L label)
void removeEdge(L label, L from)
UnmodifiableArrayList<L> listNodes(boolean isBlack)
UnmodifiableArrayList<L> listChildren(L parent)
UnmodifiableArrayList<L> listParents(L child)
L getChildByEdgeLabel(L parent, L edgeLabel)
L getParentByEdgeLabel(L child, L edgeLabel)
Object getNodeData(L label)
void setNodeData(L label, Object data)
void clear()
```

המחלקה מחזיקה מפה שמפתחותיה הם תוויות הצמתים וערכיה הם הצמתים עצמם. משיקולים זהים של גישה מהירה וניהול פשוט, נבחרה מפה. הפעולות מאפשרות גישה ושינוי של הגרף ובעיקר, שליפה מהירה ובטוחה של נתוני הורים ובנים.

ג. מימוש נוסף אפשרי ופשוט למימוש הוא לשמור פשוט רשימה של צמתים ורשימה של קשתות. מימוש זה יהיה פחות יעיל, שכן כל פעולה של הוספה והסרה או חיפוש גוררת מעבר על כל הרשימות, הן כדי למצוא את הקשת או הצומת המבוקש והן כדי לבדוק חוקיות של פעולה, לדוגמא למציאת כל הצמתים השחורים יש צורך לעבור על **כל** הרשימה של הצמתים.

ה. להלן פירוט הבדיקות שנעשו:

- בדיקה בסיסית הבודקת קריאות לכל מתודות הדרייבר ללא מקרי קצה ושגיאות
- בדיקה של הוספת צמתים במקרים מיוחדים – הוספת צומת שכבר קיימת
- הוספת קשתות במקרים מיוחדים – קשת עם תווית שקיימת עבור הבן או עבור האב, קשת בין צמתים באותו הצבע, קשת עצמית, קשת עם תווית קיימת אך מחברת בין צמתים שלהם אין קשתות עם תווית זו, קשת המחברת בין צמתים שלא קיימים
- בדיקות של התמודדות עם null בתור פרמטר
- בדיקת מחיקת קשתות על כל המצבים
- בדיקת מחיקת צמתים
- בדיקה של איפוס הגרף (מחיקת כל הקשתות והצמתים)
- בדיקה של החזרת האובייקט שמוכל בצומת

בדיקות אלו מספיקות מכיוון שבבדיקות קופסא שחורה המימוש עצמו לא נבדק אלא המפרט. לכן עבור כל מתודה נבדקו המצבים השונים האפשריים בקריאה למתודה בהתאם למתואר במפרט, יחד עם בדיקה שההתנהגות היא כמצופה מהמפרט.