שאלה 1

1. זריקה של חריגה (בפרט CloneNotSupportedException) במפרט של מתודת clone() מעידה על כך שהמחלקה לא תומכת בפעולת שכפול.  
   המחלקה Shape מממשת את הממשק Cloneable, לכן היא כן תומכת בשכפול.
2. המחלקה java.awt.Color היא immutable:
   1. היא לא מממשת את הממשק Cloneable ולא יורשת ממחלקה שמממשת אותו.
   2. אין לה מתודות setter.

ההשפעה על המחלקה Shape היא:

1. במתודות setColor(Color), getColor() לא משתמשים במתודה clone() בשביל לשכפל את הפרמטר/השדה (ב- getColor/setColor בהתאמה).
2. כשמשכפלים את Shape (ב-clone()) לא צריך ליצור שכפול של השדה color לאחר הקריאה ל-super.clone() מכיוון שזאת יוצרת shallow copy של כל השדות של המחלקה, שמספיק עבור שדות שהם immutable.

שאלה 3

1. בסעיף ב' השתמשנו בטיפוס Shape שממנו יורשות כל הצורות הגיאומטריות ובסעיף ג' רק באובייקטים שגם מממשים את הממשק Animatable.

זאת משום שכל הצורות יורשות מ-Shape, אבל לא כולן בהכרח מממשות את Animatable, ובסעיף ג' היינו צריכים לממש את האנימציה.

ניתן לבצע את הפעולות הנ"ל תודות ל- dynamic polymorphism וגם לאופרטור instanceof בג'אווה, אשר מחזיר ערך בוליאני true אם אובייקט הוא מטיפוס מסוים, או יורש ממנו, או מממש אותו (במקרה של ממשק).

1. TODO
2. אם הסטודנט מימש כמו שצריך – קבע את טיפוס המשתנה shapes שיהיה Collection (או במקרה הזה אפשר גם List), אז הוא לא יצטרך לבצע שום שינוי נוסף בקוד מכיוון שהמעבר על הצורות במיכל הוא בעזרת איטרטור, וכל המחלקות שיורשות מ-Collection (שבינהם ArrayList ו-LinkedList) צריכות לממש את הממשק Iterable.