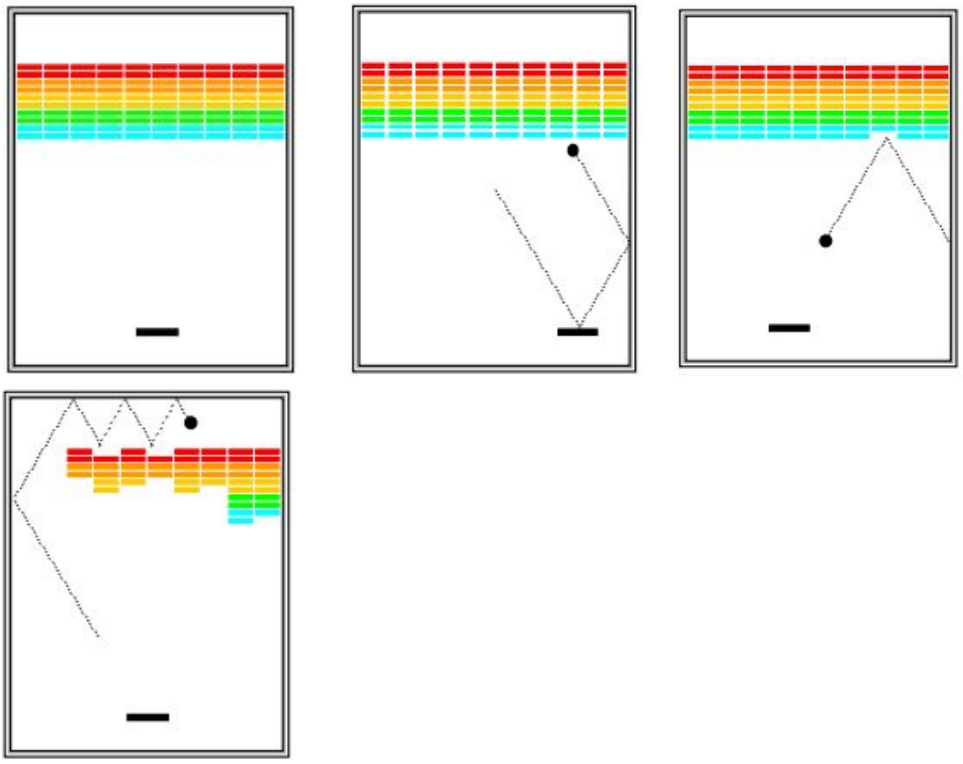
!!!Увага завдання може бути виконано групою з 2 студентів або самостійно!!!

Завдання для лабораторної

Вашим завданням буде напити класичну аркадну гру Breakout, яку розробив Стів Возняк перед заснуванням компанії Apple разом із Стівом Джобсом.

Гра Breakout



В Breakout світ виглядає, як показано на малюнках вище. Кольорові чотирикутники вгорі – цеглини, трохи більший, що знизу – ракетка. Ракетка знаходить у фіксованій позиції у вертикальній площині, але може пересуватися в горизонтальній, обмежена стінками.

Гра складається з трьох змін напрямку. Кожного разу м’яч запускається з центру вікна вниз під випадковим кутом. Далі він відбивається від ракетки та стін ігрового світу відповідно до фізичного принципу – «кут падіння дорівнює куту відбивання». Таким чином, після відбивання від ракетки та стін, м’яч має траєкторію, що показана на другому малюнку (лінія, яка показую траєкторію руху м’яча не має відображатися).

Як ви можете бачити, м’яч відбивається від однієї з нижніх цеглин, під час цього – вона зникає. Третій малюнок показує, що відбувається потім.

Гра продовжується до тих пір, поки не виконається одна з наступних умов:

1. Коли м’яч попадає на нижню стінку, що значить, що гравець його не відбив. В цьому випадку дається ще одне «життя» (ще один м’яч), але якщо вони закінчилися, то гра припиняється з програшем.
2. Коли відбита остання цеглина. В цьому випадку гравець виграв.

Після того, як певна кількість колонок розбита, відкривається простір до верхньої стінки. Коли таке трапляється, то м’яч може перелетіти на верхній рівень. На останньому малюнку наведена ілюстрація даної ситуації.

Файл starter

Початковий вміст файлу **Breakout.java** наведено на рис.1. Файл має наступні деталі:

- Імпорт файлів, які знадобляться вам для написання програми
- Визначає константи, які контролюють ігрові параметри, такі як розмірність різних об’єктів. Ваш код повинен використовувати дані константи внутрішньо, і якщо ви їх зміните, то разом з ними зміниться поведінка програми.

Успіх написання даної програми залежить від розбиття задачі на більш малі під задачі, а також, примусити їх працювати перед тим, як просуватись далі.

Установка цеглин

Перед тим як грати, вам потрібно запустити різні частинки. Таким чином, потрібно виконати метод **run,** як два виклики метода: один запускає гру, інший її виконує. Важлива частина запуску гри – створення рядків цеглин, що виглядають наступним чином:

Кількість, розмірність та відстань між цеглинами визначена константами у файлі**,** як відстань від верхньої частини вікна до першого рядка цеглин. Єдине значення, яке вам потрібно порахувати – це координата x першої колонки цеглин, що має бути відцентрована у вікні гри. Кольори також константи, які знаходяться в наступному порядку: **RED, ORANGE, YELLOW, GREEN, CYAN.**

Рис.1

```
/*
 * File: Breakout.java
 * -------------------
 * This file will eventually implement the game of Breakout.
 */

import acm.graphics.*;
import acm.program.*;
import acm.util.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Breakout extends GraphicsProgram {

/** Width and height of application window in pixels */
   public static final int APPLICATION_WIDTH = 400;
   public static final int APPLICATION_HEIGHT = 600;

/** Dimensions of game board (usually the same) */
   private static final int WIDTH = APPLICATION_WIDTH;
   private static final int HEIGHT = APPLICATION_HEIGHT;

/** Dimensions of the paddle */
   private static final int PADDLE_WIDTH = 60;
   private static final int PADDLE_HEIGHT = 10;

/** Offset of the paddle up from the bottom */
   private static final int PADDLE_Y_OFFSET = 30;

/** Number of bricks per row */
   private static final int NBRICKS_PER_ROW = 10;

/** Number of rows of bricks */
   private static final int NBRICK_ROWS = 10;

/** Separation between bricks */
   private static final int BRICK_SEP = 4;

/** Width of a brick */
   private static final int BRICK_WIDTH =
     (WIDTH - (NBRICKS_PER_ROW - 1) * BRICK_SEP) / NBRICKS_PER_ROW;

/** Height of a brick */
   private static final int BRICK_HEIGHT = 8;

/** Radius of the ball in pixels */
   private static final int BALL_RADIUS = 10;

/** Offset of the top brick row from the top */
   private static final int BRICK_Y_OFFSET = 70;

/** Number of turns */
   private static final int NTURNS = 3;

   public void run() {
      /* You fill this in, along with any subsidiary methods */
   }
}
```

Створення ракетки

Наступним кроком є створення ракетки. В нас є лише одна ракетка, яка заповнюється **GRect.** Ви, навіть, знаєте її позицію у вікні відносно нижньої границі.

Складність тут полягає у тому, щоб примусити ракетку рухатися за курсором. Тут вам потрібно звернути увагу на вісь x, оскільки по y – ракетка зафіксована. Також потрібно зробити обмеження, щоб ракетка не виходила за межі.

Створення м’яча та його рух

З однієї сторони створення м’яча досить просте завдання – заповнити **GOval.** Важче примусити його рухатися та відбиватися правильно. Тут ви вже пройшли фазу «запуску» та перейшли до другої – фази «гри». Для початку створіть м’яч та поставте його по центру вікна. Коли ви це робите, майте на увазі, що координати **GOval** не визначають центр м’яча, а його верхній лівий кут.

М’ячу потрібна швидкість, яка складається з двох компонент, що оголошуються наступним чином:

private double vx, vy;

Компоненти швидкості показують зміну позиції в кожен момент часу. Ви можете спробувати початкове значення швидкості +3,0 для **vy**. Гра буде нудною, якщо м'яч падатиме з одного і того самого місця, тому вам потрібно обрати компоненту **vx** випадковим чином.

Ось, що вам потрібно зробити:

1. Оголосіть змінну **rgen**, яка слугуватиме генератором випадкових чисел:
   private RandomGenerator rgen = RandomGenerator.getInstance();
2. Ініціалізуйте змінну **vx** наступним чином:
   vx = rgen.nextDouble(1.0, 3.0);
   if (rgen.nextBoolean(0.5)) vx = -vx;

Цей код надає змінній **vx** випадкового значення **double** в проміжку між 1.0 та 3.0, потім робить її від'ємною. Це краще, ніж наступним чином:

nextDouble(-3.0, +3.0)

що зробить падіння м'яча більш-менш прямим. Що буде дуже просто для гравця.

Коли ви все це виконаєте, вашим наступним завданням буде примусити м'яч відбиватися від стінок, ігноруючи цеглини та ракетку. Для того, щоб це зробити, вам потрібно перевірити, щоб координати м'яча не виходили за межі світу, приймаючи до уваги те, що м'яч має не нульовий розмір. Таким чином, щоб м'яч відбився від правої стінки, вам потрібно зробити так, щоб координати правої межі м'яча стали більшими за ширину вікна, інші три напрямки виконуються аналогічно.

Порахувати, що відбуватиметься далі досить просто. Якщо м'яч відбивається від верхньої або нижньої стінки, все, що вам потрібно зробити – це змінити знак **vy**. на протилежний. Так само і з **vx.**

Перевірка колізій

Припустимо, що м'яч був точкою, а не колом. Як ви визначите в цьому випадку, що він зіткнувся з іншим об'єктом?

public GObject getElementAt(double x, double y)

який отримує позицію у вікні та повертає графічний об'єкт на позицію. Якщо немає жодного графічного об'єкта, тоді **getElementAt** повертає спеціальну константу null. Якщо їх більше, ніж один, то **getElementAt** завжди обирає того, що знаходиться найближче до вершини стеку.

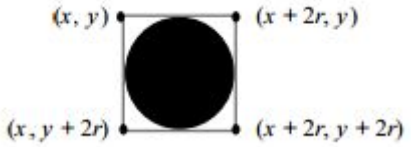
Що трапиться, якщо ви викличете наступне:

getElementAt(x, y),

де x та y координати м'яча? Якщо точка **(x, y)** знаходиться під об'єктом, то цей виклик поверне графічний об'єкт, з яким зіткнувся м'яч. Якщо в цій точці немає об'єктів, то ви отримаєте значення **null.**

Але, на жаль, м'яч не є точкою. Він займає фізичну площу і тому може відбиватися від інших об'єктів, навіть, якщо його центр цього не робить. Найлегше, що можна зробити в даній ситуації – це обрати декілька точок за межами м'яча і подивитися, чи відбиваються ці точки від інших предметів. Коли ці точки будуть взаємодіяти з іншими об'єктами, то ви можете сказати, що це відбився від перешкоди ваш м'яч.

В вашій реалізації, найлегшим буде перевірити чотири точки по кутах квадрату, в який вписано ваш м'яч. Пам'ятайте, що **GOval** визначається межами прямокутника, якщо верхній лівий кут прямокутника знаходиться в точці (x, y), інші кути будуть знаходитися в локаціях, що показані нижче:



Ці точки мають перевагу, знаходячись поза межами м'яча – це означає, що **getElementAt** не може повернути в результаті сам м'яч, тим не менш, достатньо близько для того, щоб з'явилися колізії. Таким чином для кожної з цих чотирьох точок вам потрібно:

1. Викликати **getElementAt** на даній локації, і подивитися, чи там щось є.
2. Якщо значення, що ви отримали не **null**, тоді вам не потрібно шукати далі, і ви можете взяти це значення як **GObject,** з яким трапилася колізія.
3. Якщо **getElementAt** повернуло **null,** то вам потрібно перевірити наступний кут.
4. Якщо ви перевірили всі чотири кути і не знайшли колізій, то їх не існує.

Буде дуже корисно написати наступну секцію як окремий метод:

private GObject getCollidingObject()

який повертає об'єкт, що був залучений під час колізії, якщо він не **null.** Далі ви можете використовувати його в оголошенні:

GObject collider = getCollidingObject();

Що надає значення змінній **collider**.

Починаючи з цього моменту вам залишається придумати, що робити, коли трапляється колізія. У вас є лише дві можливості. Перша, об'єкт, що ви отримуєте можна відбити, перевірити цю можливість наступним чином:

if (collider == paddle) . . .

якщо це ракетка, то вам потрібно відбити м'яч. Якщо це не ракетка, то це цеглина. При цьому вам потрібно прибрати її. Для того, щоб це виконати вам потрібно викликати метод **remove**.

Закінчення

Якщо ви сюди дісталися, то вже виконали найважчу частину роботи. Але, залишилося ще декілька деталей, що потрібно взяти до уваги:

- Якщо м'яч попадає на нижню стінку. В прототипі, що ви побудували, м'яч відбивається від неї так само, як і від інших, але це робить гру фактично безпрограшною. Ви маєте модифікувати структуру вашого циклу так, що він

тестуватиме попадання м’яча на нижню стінку і буде закінчувати його перериванням.

- Також вам потрібно перевірити ще одне переривання, коли зникає остання цеглина. Найлегшим способом буде зробити так, щоб ваша програма рахувала, скільки цілих цеглин ще залишилось. Коли досягається значення 0, то ви переривається цикл. Також можна додати feedback гравцю.
- Ви експериментували з налаштуваннями, що контролюють швидкість вашої програми. Як довго вам потрібно робити паузу в циклі, щоб оновити м’яч? Вам потрібно змінювати значення швидкості, щоб гра була цікавішою?
- Ви маєте протестувати вашу програми, щоб переконатися, що вона дійсно працює. Пограйте в неї трохи. Якщо ви вважаєте, що все працює, перевірте наступне: перед тим, як м’яч пересікатиме лінію ракетки, пересуньте її настільки швидко, щоб вона збила м’яч, а не навпаки. Все працює? Чи здається, ніби м’яч «приклеївся» до ракетки? Якщо у вас виникла ця помилка, спробуйте розібратися чому саме, та як її виправити.

Стратегія і тактика

Ось деякі поради до цього завдання:

- *Почніть виконувати його якомога раніше.*
- *Розділіть програму на частини.* Не намагайтеся зробити так, щоб все запрацювало одразу. Спочатку виконайте малі частинки, і переконайтеся, що вони працюють, перед тим, як перейти до наступної фази.
- *Зробіть розклад виконання завдання та слідуйте ньому.*
- *Не намагайтеся розширити можливості вашої програми, поки не запрацює основна її частина.* Наступна секція описує декілька шляхів для розширення можливостей програми.

Можливе розширення

- *Додайте звуки*. Коли м’яч відбивається від стінки або цеглини. Початковий проект містить аудіо файл **bounce.au,** що містить цей звук. Ви можете завантажити його наступним чином:

  AudioClip bounceClip = MediaTools.loadAudioClip("bounce.au");

  Далі запустіть його викликом

  bounceClip.play();

- *Додайте повідомлення.* Повідомлення про виграш або програш. Це лише об’єкти **GLabel** яки ви можете додати або видалити в певному місці.
- *Покращити керування відбиванням.* Буде цікавіше, якщо гравець зможе відбивати м’яч різними частинами ракетки.
- *Додайте “kicker”.* Якщо гра стає нудною, можна підняти її швидкість.

- *Додайте рахунок очок.* Наприклад, за збиті цеглини, причому в залежності від кольорів.
- *Використовуйте вашу уяву.* Що ви завжди хотіли, щоб було в цій грі?