**Арканоид(2)**

Главному классу (Arkanoid) нужно будет хранить информацию о размерах поля, где будут происходить все действия.

Поэтому:  
а) добавь в класс Arkanoid два приватных поля width(ширина) и height(высота) типа int  
б) создай для них публичные геттеры и сеттеры

### Арканоид(3)

Пожалуй, пора добавить конструктор в класс **Arkanoid**.  
Конструктор должен принимать два параметра типа int (width и height) и корректно устанавливать значения соответствующих полей класса.

### Арканоид(4)

Также классу **Arkanoid** нужно будет хранить ссылку на шарик (**Ball**), «***подставку***» **Stand** и список «***кирпичей***«.

**Надо:**  
а) создать в классе **Arkanoid** два private поля: ball типа **Ball** и stand типа Stand  
б) добавить private поле bricks типа **List**<Brick>  
в) добавь для них публичные **геттеры** и **сеттеры**

### Арканоид(5)

Чего еще не хватает классу **Arkanoid**?  
Во-первых ему нужен метод run(), в котором будет описана основная логика программы.  
Еще нужен метод move() — который будет двигать на один шаг все объекты требующие движения.  
Создай методы run() и move().

Еще нам понадобится **НЕ** приватная статическая переменная game типа **Arkanoid**, которая будет хранить ссылку на созданный экземпляр класса **Arkanoid**.

**Выглядеть это должно примерно так:**  
static Arkanoid game;

### Арканоид(6)

У нас будут «***кирпичи***«, «***шарик***» и «***подставка***» и у них будет много общего.  
Они будут перемещаться по полю и отрисовываться.  
Значит у них у всех будут координаты и размер.  
А еще методы move() — для перемещения и draw() для отрисовки.

Есть интересное предложение: давай введем один базовый класс для все объектов.  
Пусть это будет класс **BaseObject**.  
А классы **Ball**, **Stand**, **Brick** от него наследуются.

Создай класс **BaseObject** и добавь его родителем к классам **Ball**, **Stand**, **Brick**

Еще нам понадобится класс **Canvas**  
Он будет ответственным за «***отрисовку***» объектов.  
С помощью его они будут отрисовывать себя.  
Вернее даже на нем, но детали я сообщу позднее.

**Создай и этот класс.**

**Арканоид(7)**

Теперь перейдем к классу BaseObject.  
Я хочу сделать несколько предложений.

Во-первых для простоты считать все объекты у нас будут круглыми.  
Нет, отрисовывать их мы будем фигурными, как и раньше.  
А вот при расчетах из взаимодействия исходить из того, что они круглые.  
Так — гораздо проще.

Во-вторых. Пусть координаты объектов и радиус будут вещественными числами.  
Это придаст плавность движениям и точность всем вычислениям.  
А при отрисовке мы будем их округлять.

Итак:  
а) Добавь в класс BaseObject приватные поля x (double), y (double), radius (double).  
б) Добавь геттеры и сеттеры.  
в) Добавь конструктор BaseObject(double x, double y, double radius).  
г) Пройдись по все классам-наследникам и поправь у них конструкторы.

Если вы пользуйтесь Intellij IDEA — Alt+Insert вам в помощь.

Требования:  
1. В классе BaseObject должно быть создано приватное поле x типа double.  
2. В классе BaseObject должно быть создано приватное поле y типа double.  
3. В классе BaseObject должно быть создано приватное поле radius типа double.  
4. В классе BaseObject должен быть создан корректно работающий конструктор с тремя параметрами типа double (x, y, radius).  
5. В классе Ball должен быть создан корректно работающий конструктор с тремя параметрами типа double (x, y, radius).  
6. В классе Brick должен быть создан корректно работающий конструктор с тремя параметрами типа double (x, y, radius).  
7. В классе Stand должен быть создан корректно работающий конструктор с тремя параметрами типа double (x, y, radius).  
8. В классе BaseObject должен быть создан публичный геттер для поля x.  
9. В классе BaseObject должен быть создан публичный геттер для поля y.  
10. В классе BaseObject должен быть создан публичный геттер для поля radius.  
11. В классе BaseObject должен быть создан публичный сеттер для поля x.  
12. В классе BaseObject должен быть создан публичный сеттер для поля y.  
13. В классе BaseObject должен быть создан публичный сеттер для поля radius.

### Арканоид(8)

Но и это еще не все.  
Классу BaseObject нужны еще методы.  
Пока это будут пустые методы draw(Canvas canvas) и move().  
Классы-наследники должны будут переопределить их у себя и реализовать необходимую функциональность (пока можешь оставить их пустыми, просто чтобы код компилировался).  
Объяви эти методы, но сделай их абстрактными.  
Также сделай абстрактным сам класс BaseObject.

А еще нам нужно будет определять попал шарик в кирпич или в подставку.  
**Это будем делать так:**  
В этом же классе, создадим специальный метод: boolean isIntersec(BaseObject **o**)  
Он будет определять — «***пересеклись***» объекты или нет. Если пересеклись — возвращать **true**, если нет — **false**.

Т.к. объекты мы условно считаем кругами, то предлагаю такую формулу взаимодействия  
Если центр круга одного объекта попал в круг другого, то будем считать, что они столкнулись.

**Или еще проще:**  
дистанция\_между\_объектами <= max (радиус\_первого\_объекта, радиус\_второго\_объекта)

Требования:  
1. Класс BaseObject должен быть абстрактным.  
2. В классе BaseObject должен быть объявлен абстрактный метод draw() c одним параметром типа Canvas.  
3. В классе BaseObject должен быть объявлен абстрактный метод move() без параметров.  
4. В классе BaseObject должен быть реализован метод isIntersec() с одним параметром типа BaseObject, возвращающий true в случае, если текущий объект пересекается с анализируемым.

### Арканоид(9)

Теперь займемся классом **Canvas**.  
Он у нас будет содержать матрицу (***двумерный массив***), куда мы будем «***рисовать***«.  
У матрицы есть ширина и высота.  
А еще будем в ней хранить не **числа** (int), а **символы** (char).

**Надо:**  
а) Добавить в класс два приватных поля width и height.  
б) Добавить в класс приватное поле matrix (char[][]).  
в) Добавить конструктор с двумя параметрами типа int (width и height), поле matrix должно быть инициализировано пустым массивом размерностью **[**height**+2][**width**+2]**.  
г) Добавить геттеры и сеттеры для всех приватных полей класса.

### Арканоид(10)

Что мы будем делать с **Canvas**?  
Мы будем рисовать на нем (в его матрице).  
Поэтому нам понадобятся два метода  
void setPoint(double **x**, double **y**, char **c**)  
void drawMatrix(double **x**, double **y**, int[][] **matrix**, char **c**)

Первый метод — setPoint будет «ставить точку в координатах x**,**yцветомc«.

**В методе надо:**  
а) округлить x и y до целых чисел  
б) занести в matrix[y][x] значение с  
в) ничего не делать, если x < **0** или y < 0 или y > matrix.length или x > matrix[0].length

Второй метод — drawMatrix копирует переданную ему картинку (***матрицу***) в матрицу **Canvas**.  
И не просто копирует, а начиная с координат x, y

**В методе надо:**  
а) с помощью двух вложенных циклов пройтись по всем ячейкам переданной картинки  
б) если значение ячейки элемента **[i][j]** полученной матрицы не равно **0**, то покрасить в матрице объекта **Canvas** точку **(**x**+j,** y**+i)** в цвет c:  
setPoint(x+j, y+i, c)

Требования:  
1. В классе Canvas должен быть создан метод void setPoint(double x, double y, char c).  
2. В классе Canvas должен быть создан метод void drawMatrix(double x, double y, int[][] matrix, char c).  
3. Поведение метода setPoint должно соответствовать условию задачи.  
4. Поведение метода drawMatrix должно соответствовать условию задачи.