## **Задание 2**

Распакуйте архив “Задание 2.zip“ в Ваш рабочий каталог. В Вашем проекте после этого должны появиться классы TestDesktop, TestPoint, TestRectButton, TestRoundButton и TestWindowFactory в каталоге test/java в пакете net.thumbtack.school.windows.v1

Создайте пакет net.thumbtack.school.windows.v1 в каталоге main/java и добавьте туда следующие классы и методы в них. Обращаем Ваше внимание, что все поля классов должны иметь атрибут private.

Во всех классах ось X направлена вправо, ось Y - **вниз**.

**Desktop**

Экран компьютера. Координаты отсчитываются от левой верхней точки.

1. **public Desktop(int width, int height)**

Создает Desktop по значениям ширины и высоты.

1. **public Desktop()**

Создает классический VGA Desktop с размерами 640\*480

1. **public int getWidth()**

Возвращает ширину экрана

1. **public int getHeight()**

Возвращает высоту экрана

1. **public int getArea()**

Возвращает площадь экрана (иными словами, количество пикселей)

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**Point**

Точка на экране. Можно воспользоваться кодом из класса Point2D, приводимого в лекции.

1. **public Point(int x, int y)**

Создает Point с заданными координатам

1. **public Point()**

Создает Point с координатам (0,0)

1. **public Point(Point point)**

Создает point с координатами теми же, что и у передаваемого Point

1. **public int getX()**

Возвращает x-координату Point

1. **public int getY()**

Возвращает y-координату Point

1. **public void setX(int x)**

Устанавливает новую x-координату Point

1. **public void setY(int y)**

Устанавливает новую y-координату Point

1. **public void moveTo(int x, int y)**

Перемещает точку в новые координаты x, y

1. **public void moveRel(int dx, int dy)**

Перемещает точку в новые координаты, смещая ее на (dx, dy) от текущего положения.

1. **public boolean isVisibleOnDesktop(Desktop desktop)**

Определяет, находится ли точка в пределах Desktop

1. **public boolean isNotVisibleOnDesktop(Desktop desktop)**

Определяет, находится ли точка вне пределов Desktop

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**RectButton**

Нажимная прямоугольная кнопка. Для кнопки определено 2 состояния - активна (можно нажать) и пассивна (серого цвета, нажать нельзя). Предполагается, что всегда будут передаваться допустимые координаты, то есть при создании или изменении всегда будет выполняться : левая точка не правее правой, верхняя точка не ниже нижней.

1. **public RectButton(Point topLeft, Point bottomRight, boolean active)**

Создает RectButton по координатам углов - левого верхнего и правого нижнего, и флагу активности. Обращаем внимание на то, что обе точки входят в кнопку, так что если создать кнопку с topLeft.equals(bottomRight), то будет создана кнопка ширины и высоты 1.

1. **public RectButton(int xLeft, int yTop, int width, int height, boolean active)**

Создает RectButton по координатам левого верхнего угла, ширине, высоте и флагу активности.

1. **public RectButton(Point topLeft, Point bottomRight)**

Создает активную RectButton по координатам углов - левого верхнего и правого нижнего.

1. **public RectButton(int xLeft, int yTop, int width, int height)**

Создает активную RectButton по координатам левого верхнего угла, ширине и высоте.

1. **public Point getTopLeft()**

Возвращает левую верхнюю точку RectButton.

1. **public Point getBottomRight()**

Возвращает правую нижнюю точку RectButton.

1. **public boolean isActive()**

Возвращает true, если кнопка активна, иначе false.

1. **public void setTopLeft(Point topLeft)**

Устанавливает левую верхнюю точку RectButton.

1. **public void setBottomRight(Point bottomRight)**

Устанавливает правую нижнюю точку RectButton.

1. **public void setActive(boolean active)**

Устанавливает состояние активности RectButton.

1. **public int getWidth()**

Возвращает ширину RectButton.

1. **public int getHeight()**

Возвращает высоту RectButton.

1. **public void moveTo(int x, int y)**

Передвигает RectButton так, чтобы левый верхний угол его оказался в точке (x, y).

1. **public void moveTo(Point point)**

Передвигает RectButton так, чтобы левый верхний угол его оказался в точке point.

1. **public void moveRel(int dx, int dy)**

Передвигает RectButton на (dx, dy).

1. **public void resize(double ratio)**

Изменяет ширину и длину RectButton в ratio раз при сохранении координат левой верхней точки. Дробная часть вычисленной длины или ширины отбрасывается. Если при таком изменении длина или ширина какой-то из сторон окажется меньше 1, то она принимается равной 1.

1. **public boolean isInside(int x, int y)**

Определяет, лежит ли точка (x, y) внутри RectButton. Если точка лежит на стороне, считается, что она лежит внутри.

1. **public boolean isInside(Point point)**

Определяет, лежит ли точка point внутри RectButton. Если точка лежит на стороне, считается, что она лежит внутри.

1. **public boolean isIntersects(RectButton rectButton)**

Определяет, пересекается ли RectButton с другим RectButton. Считается, что кнопки пересекаются, если у них есть хоть одна общая точка.

1. **public boolean isInside(RectButton rectButton)**

Определяет, лежит ли RectButton целиком внутри текущего RectButton.

1. **public boolean isFullyVisibleOnDesktop(Desktop desktop)**

Определяет, верно ли, что вся RectButton находится в пределах Desktop.

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**RoundButton**

Нажимная круглая кнопка. Для кнопки определено 2 состояния - активна (можно нажать) и пассивна (серого цвета, нажать нельзя).

1. **public RoundButton(Point center, int radius, boolean active)**

Создает RoundButton по координатам центра, значению радиуса и флагу активности.

1. **public RoundButton(int xCenter, int yCenter, int radius, boolean active)**

Создает RoundButton по координатам центра, значению радиуса и флагу активности.

1. **public RoundButton(Point center, int radius)**

Создает активную RoundButton по координатам центра и значению радиуса.

1. **public RoundButton(int xCenter, int yCenter, int radius)**

Создает активную RoundButton по координатам центра и значению радиуса.

1. **public Point getCenter()**

Возвращает центр RoundButton.

1. **public int getRadius()**

Возвращает радиус RoundButton.

1. **public boolean isActive()**

Возвращает true, если кнопка активна, иначе false.

1. **public void moveTo(int x, int y)**

Передвигает RoundButton так, чтобы центр его оказался в точке (x, y).

1. **public void moveTo(Point point)**

Передвигает RoundButton так, чтобы центр его оказался в точке point.

1. **public void setCenter(int x, int y)**

Устанавливает центр RoundButton.

1. **public void setRadius(int radius)**

Устанавливает радиус RoundButton.

1. **public void setActive(boolean active)**

Устанавливает состояние активности RoundButton.

1. **public void moveRel(int dx, int dy)**

Передвигает RoundButton на (dx, dy).

1. **public void resize(double ratio)**

Изменяет радиус RoundButton в ratio раз, не изменяя центра. Дробная часть вычисленного таким образом радиуса отбрасывается. Если вычисленный радиус окажется меньше 1, то он принимается равным 1.

1. **public boolean isInside(int x, int y)**

Определяет, лежит ли точка (x, y) внутри RoundButton. Если точка лежит на окружности, считается, что она лежит внутри. В этом методе мы пренебрегаем пиксельной структурой изображения и рассматриваем RoundButton как математический круг.

1. **public boolean isInside(Point point)**

Определяет, лежит ли точка point внутри RoundButton. Если точка лежит на окружности, считается, что она лежит внутри. В этом методе мы пренебрегаем пиксельной структурой изображения и рассматриваем RoundButton как математический круг.

1. **public boolean isFullyVisibleOnDesktop(Desktop desktop)**

Определяет, верно ли, что вся RoundButton находится в пределах Desktop.

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**WindowFactory**

Класс, создающий окна (фабрика окон)

1. **public static RectButton createRectButton(Point leftTop, Point rightBottom, boolean active)**

Создает RectButton по координатам точек и флагу активности.

1. **public static RoundButton createRoundButton(Point center, int radius, boolean active)**

Создает RoundButton по координатам центра, значению радиуса и флагу активности.

1. **public static int getRectButtonCount()**

Возвращает количество RectButton, созданных с помощью метода createRectButton.

1. **public static int getRoundButtonCount()**

Возвращает количество RoundButton, созданных с помощью метода createRoundButton.

1. **public static int getWindowCount()**

Возвращает общее количество окон (RectButton и RoundButton), созданных с помощью методов WindowFactory.

1. **public static void reset()**

Устанавливает количество всех окон, созданных с помощью методов WindowFactory, равным 0 (иными словами, реинициализирует фабрику).

Проверьте работу тестов в консольном окне, запишите все классы на сервер (не забудьте изменить текст сообщения в git commit!) и убедитесь, что на сервере все тесты также проходят успешно (см. Занятие 1, п.15-19)