## **Задание 4**

Распакуйте архив “Задание 4.zip” в Ваш рабочий каталог. В Вашем проекте после этого должны появиться классы TestComboBox, TestRadioButton и TestRectButton3D в каталоге test/java в пакете net.thumbtack.school.windows.v2

Используя разработанные в заданиях 2-3 классы RectButton, RoundButton, ListBox, Point, Desktop создайте в пакете net.thumbtack.school.windows.v2классы, указанные ниже**.** При разработке этих классов настоятельно рекомендуется подумать, какие методы должны быть в них действительно написаны, а какие можно наследовать от базового класса.

Напоминаем, что все поля классов должны иметь атрибут private.

Везде оси X и Y направлены так же, как и в задании 2, ось Z - перпендикулярно экрану компьютера.

**RectButton3D**

Нажимная прямоугольная кнопка , имеющая трехмерный вид. 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 выше нижней, высота по оси Z больше 0.

1. **public RectButton3D(Point topLeft, Point bottomRight, boolean active, String text, int zHeight)**

Создает RectButton3D по координатам углов - левого верхнего и правого нижнего, высоте zHeight, строке на кнопке text 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 RectButton3D(int xLeft, int yTop, int width, int height, boolean active, String text, int zHeight)**

Создает RectButton3D по координатам левого верхнего угла, ширине, высоте, строке на кнопке text, высоте zHeight и флагу активности.

1. **public RectButton3D(Point topLeft, Point bottomRight, String text, int zHeight)**

Создает активную RectButton3D по координатам углов - левого верхнего и правого нижнего, строке на кнопке text и высоте zHeight.

1. **public RectButton3D(int xLeft, int yTop, int width, int height, String text, int zHeight)**

Создает активную RectButton3D по координатам левого верхнего угла, ширине, строке на кнопке text и высоте zHeight.

1. **public Point getTopLeft()**

Возвращает левую верхнюю точку RectButton3D.

1. **public Point getBottomRight()**

Возвращает правую нижнюю точку RectButton3D.

1. **public boolean isActive()**

Возвращает true, если кнопка активна, иначе false

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

Устанавливает левую верхнюю точку RectButton3D.

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

Устанавливает правую нижнюю точку RectButton3D.

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

Устанавливает состояние активности RectButton3D.

1. **public int getWidth()**

Возвращает ширину RectButton3D.

1. **public int getHeight()**

Возвращает высоту RectButton3D.

1. **public int getZHeight()**

Возвращает высоту RectButton3D по оси Z.

1. **public void setZHeight(int zHeight)**

Устанавливает высоту RectButton3D по оси Z.

1. **public String getText()**

Возвращает текст окна

1. **public void setText(String text)**

Устанавливает текст окна

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

Передвигает RectButton3D так, чтобы левый верхний угол его оказался в точке (x, y).

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

Передвигает RectButton3D так, чтобы левый верхний угол его оказался в точке point.

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

Передвигает RectButton3D на (dx, dy).

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

Изменяет ширину и длину RectButton3D 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. Высота не изменяется.

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

Определяет, лежит ли точка (x, y) внутри RectButton3D. Если точка лежит на стороне, считается, что она лежит внутри.

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

Определяет, лежит ли точка point внутри RectButton3D. Если точка лежит на стороне, считается, что она лежит внутри.

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

Определяет, пересекается ли RectButton3D с другим RectButton или RectButton3D. 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 RectButton3D. Поскольку RectButton не имеет z-высоты, она при сравнении не учитывается.

1. **public boolean isInside(RectButton3D rectButton3D)**

Определяет, лежит ли RectButton3D целиком внутри текущего RectButton3D. Высота учитывается.

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

Определяет, верно ли, что вся RectButton3D находится в пределах Desktop.

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**RadioButton**

Круглая кнопка, внутри которой может быть точка. Несколько таких кнопок, объединенных в группу, используются для выбора одного из нескольких вариантов. Для кнопки определено 2 состояния - активна (можно нажать) и пассивна (серого цвета, нажать нельзя).

1. **public RadioButton(Point center, int radius, boolean active, String text, boolean checked)**

Создает RadioButton по координатам центра, значению радиуса, флагу активности, тексту и состоянию.

1. **public RadioButton(int xCenter, int yCenter, int radius, boolean active, String text, boolean checked)**

Создает RadioButton по координатам центра, значению радиуса, флагу активности, тексту и состоянию.

1. **public RadioButton(Point center, int radius, String text, boolean checked)**

Создает активную RadioButton по координатам центра, значению радиуса, тексту и состоянию.

1. **public RadioButton(int xCenter, int yCenter, int radius, String text, boolean checked)**

Создает активную RadioButton по координатам центра, значению радиуса, тексту и состоянию.

1. **public Point getCenter()**

Возвращает центр RadioButton.

1. **public int getRadius()**

Возвращает радиус RadioButton.

1. **public boolean isActive()**

Возвращает true, если кнопка активна, иначе false.

1. **public boolean isChecked()**

Возвращает true, если точка внутри кнопки установлена, иначе false.

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

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

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

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

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

Передвигает RadioButton на (dx, dy).

1. **public void setCenter(Point center)**

Устанавливает центр RadioButton.

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

Устанавливает радиус RadioButton.

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

Устанавливает состояние активности RadioButton.

1. **public void setChecked(boolean checked)**

Устанавливает или убирает точку внутри RadioButton.

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

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

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

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

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

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

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

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

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

**ComboBox**

Прямоугольное окно, содержащее в себе список строк, одна из которых является выделенной (ее может и не быть). Для ComboBox определено 2 состояния - активен (изображается черным цветом) и пассивен (серым цветом).

Предполагается, что всегда будут передаваться допустимые аргументы, то есть при создании или изменении всегда будет выполняться : левая точка не правее правой, верхняя точка не выше нижней, номер выбранной строки не больше значения (количество строк -1), если массив строк отсутствует, номер выбранной строки будет обязательно null. Тем не менее в результате нескольких вызовов методов экземпляр класса может перейти в неконсистентное состояние, например, если был установлен номер выбранной строки, а после этого массив строк устанавливается в null. В рамках настоящего Задания мы не будем заниматься этими вопросами, отложим их до Задания 6.

Обращаем Ваше внимание, что ComboBox должен иметь свою собственную копию массива, а не хранить в себе переданный массив, (подробности см. в описании класса ListBox).

1. **public ComboBox(Point topLeft, Point bottomRight, boolean active, String[] lines, Integer selected)**

Создает ComboBox по координатам углов - левого верхнего и правого нижнего, флагу активности, набору строк и номеру выделенной строки. Если выделенной строки нет, в качестве “selected” передается null. Обращаем внимание на то, что обе точки входят в ComboBox, так что если создать ComboBox с topLeft.equals(bottomRight), то будет создан ComboBox ширины и высоты 1. Параметр lines может быть null.

1. **public ComboBox(int xLeft, int yTop, int width, int height, boolean active, String[] lines, Integer selected)**

Создает ComboBox по координатам левого верхнего угла, ширине, высоте, флагу активности, набору строк и номеру выделенной строки. Если выделенной строки нет, в качестве “selected” передается null. Параметр lines может быть null.

1. **public ComboBox(Point topLeft, Point bottomRight, String[] lines, Integer selected)**

Создает активный ComboBox по координатам углов - левого верхнего и правого нижнего, набору строк и номеру выделенной строки. Если выделенной строки нет, в качестве “selected” передается null. Параметр lines может быть null.

1. **public ComboBox(int xLeft, int yTop, int width, int height, String[] lines, Integer selected)**

Создает активный ComboBox по координатам левого верхнего угла, ширине, высоте, набору строк и номеру выделенной строки. Если выделенной строки нет, в качестве “selected” передается null. Параметр lines может быть null.

1. **public Point getTopLeft()**

Возвращает левую верхнюю точку ComboBox.

1. **public Point getBottomRight()**

Возвращает правую нижнюю точку ComboBox.

1. **public boolean isActive()**

Возвращает true, если ComboBox активен, иначе false.

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

Устанавливает левую верхнюю точку ComboBox.

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

Устанавливает правую нижнюю точку ComboBox.

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

Устанавливает состояние активности ComboBox.

1. **public int getWidth()**

Возвращает ширину ComboBox.

1. **public int getHeight()**

Возвращает высоту ComboBox.

1. **public String[] getLines()**

Возвращает копию набора строк ComboBox.

1. **public void setLines(String[] lines)**

Устанавливает набор строк ComboBox.

1. **public Integer getSelected()**

Возвращает номер выбранной строки ComboBox.

1. **public void setSelected(Integer selected)**

Устанавливает номер выбранной строки ComboBox.

1. **public String[] getLinesSlice(int from, int to)**

Возвращает набор строк ComboBox, начиная со строки “from” и до строки (“to”-1) включительно. Если в ComboBox строк меньше, чем “to”, возвращает строки от “from” и до конца. Гарантируется, что “from” < “to”. Если массив строк равен null, возвращает null.

1. **public String getLine(int index)**

Возвращает строку с номером index. Если строки с таким номером нет или массив строк равен null, возвращает null.

1. **public void setLine(int index, String line)**

Заменяет строку с номером index. Если строки с таким номером нет или массив строк равен null,, ничего не делает.

1. **public Integer findLine(String line)**

Ищет первую совпадающую с line строку в массиве строк ComboBox. Если строка найдена, возвращает ее индекс, в противном случае возвращает null.

1. **public void reverseLineOrder()**

Переворачивает массив строк ComboBox., то есть делает 0-ю строку - последней, первую - предпоследней и т.д. Если массив строк равен null, не делает ничего.

1. **public void reverseLines()**

Переворачивает каждую строку в массиве строк ComboBox. Если массив строк равен null, не делает ничего.

1. **public void duplicateLines()**

Заменяет массив строк на новый массив, вместо каждой строки вставлены две копии ее. Если массив строк равен null, не делает ничего.

1. **public void removeOddLines()**

Заменяет массив строк на новый массив, в котором каждая нечетная исходная строка удалена. Если массив строк равен null или содержит только одну строку, не делает ничего.

1. **public boolean isSortedDescendant()**

Возвращает true, если массив строк строго упорядочен по убыванию, иначе false. Если массив строк равен null, возвращает true.

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

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

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

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

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

Передвигает ComboBox на (dx, dy).

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

Изменяет ширину и длину ComboBox 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 ComboBox. Если точка лежит на стороне, считается, что она лежит внутри.

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

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

1. **public boolean isIntersects(ComboBox comboBox)**

Определяет, пересекается ли ComboBox с другим ComboBox. Считается, что ComboBox’ы пересекаются, если у них есть хоть одна общая точка.

1. **public boolean isInside(ComboBox comboBox)**

Определяет, лежит ли ComboBox целиком внутри текущего ComboBox.

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

Определяет, верно ли, что весь ComboBox находится в пределах Desktop.

1. **методы equals и hashCode.**

Не пишите эти методы сами, используйте средства IDEA.

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)