## **Задание 5**

Распакуйте архив “Задание 5.zip” в Ваш рабочий каталог. В Вашем проекте после этого должны появиться классы классы TestComboBox, TestCursor, TestDesktop, TestListBox, TestPoint, TestRadioButton, TestRectButton, TestRectButton3D, TestRectWindow, TestRoundButton, TestRoundWindow, TestTypes, TestWindow и TestWindowFactory в каталоге test/java в пакете net.thumbtack.school.windows.v3

На этом занятии мы создадим третью версию наших классов окон.

Создайте пакет net.thumbtack.school.windows.v3 в каталоге main/java и скопируйте в него все классы из пакета net.thumbtack.school.windows.v2. Не забудьте изменить в каждом классе package!

Создайте в пакете net.thumbtack.school.windows.v3 следующие классы

* class Window. Абстрактный базовый класс - superclass для всех окон
* class RectWindow. Наследник от Window. Абстрактный базовый класс для всех прямоугольных окон
* class RoundWindow. Наследник от Window. Абстрактный базовый класс для всех круглых окон

Перенесите в класс Window те поля и методы, которые являются общими для всех окон. Перенесите в класс RectWindow те поля и методы, которые не являются общими для всех окон, но являются общими для всех прямоугольных окон. Аналогично перенесите в класс RoundWindow те поля и методы, которые являются общими для всех круглых окон.

Если какой-то из этих методов имеет одну и ту же сигнатуру (список параметров и тип результата) и реализацию для всех классов - наследников, **как существующих, так и тех, которые могут быть впоследствии созданы**) просто перенесите этот метод в родительский класс (Window или RectWindow или RoundWindow соответственно). Если метод имеет одну и ту же сигнатуру в разных классах, но реализации его должны в классах-наследниках отличаться, опишите этот метод в родительском классе (Window или RectWindow или RoundWindow соответственно) как абстрактный, а в классах-наследниках напишите его реализацию. Сделайте классы окон наследниками от RectWindow или RoundWindow. Сейчас у Вас не должно быть неабстрактных классов - непосредственных наследников от класса Window, хотя не исключено, что когда-то они появятся.

Обращаем внимание, что поля во всех классах должны иметь атрибут private, а не protected. Для доступа к полям из потомков нужно использовать геттеры и сеттеры.

Поскольку все окна могут передвигаться на Desktop и менять свой размер, создайте 2 интерфейса.

Добавьте в пакет net.thumbtack.school.windows.v3 интерфейс Movable, описав в нем следующие методы

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

Передвигает окно так, так, чтобы его базовая точка (левый верхний угол или центр соответственно) оказалась в точке (x,y)

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

Передвигает окно так, так, чтобы его базовая точка (левый верхний угол или центр соответственно) оказалась в точке point

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

Передвигает окно на (dx, dy)

Добавьте в пакет net.thumbtack.school.windows.v3 интерфейс Resizable, описав в нем следующий метод

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

Изменяет размер окна в ratio раз.

Сделайте , чтобы Ваши классы имплементировали эти интерфейсы.

Пакет net.thumbtack.school.windows.v3 теперь содержит конкретные, абстрактные классы и интерфейсы. Имеет смысл создать в нем подпакеты.

Создайте подпакет net.thumbtack.school.windows.v3.iface и переместите туда интерфейсы Movable и Resizable.

Создайте подпакет net.thumbtack.school.windows.v3.base и переместите туда классы Window, RectWindow и RoundWindow.

**Cursor**

Курсор - не окно, Он может передвигаться, но менять размеры не может. В нашем классе эти размеры даже не будут храниться.

Создайте класс Cursor, поместив его в пакет net.thumbtack.school.windows.v3.cursors. В классе должны быть следующие конструкторы и методы

1. **public Cursor(int x, int y, int cursorForm)**

Создает курсор указанной формы. Мы пока не будем конкретизировать понятие вида курсора, а просто будем считать, что имеются различные формы курсоров, каждая форма имеет свой номер, нумерация произвольная. Курсор помещается в точку (x,y).

1. **public Cursor(Point point, int cursorForm)**

Создает курсор указанной формы. Курсор помещается в точку point.

1. **public int getCursorForm()**

Возвращает форму курсора.

1. **public void setCursorForm(int cursorForm)**

Устанавливает форму курсора.

1. **public int getX()**

Возвращает x-координату курсора.

1. **public int getY()**

Возвращает y-координату курсора.

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

Устанавливает x-координату курсора.

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

Устанавливает y-координату курсора.

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

Перемещает курсор в точку (x,y).

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

Перемещает курсор в точку point.

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

Перемещает курсор на (dx,dy).

Оказывается, метод public void moveTo(Point point) во всех классах имеет один и тот же вид (если у Вас это не так, подумайте, как сделать, чтобы это было именно так!). Перенесем поэтому его имплементацию непосредственно в интерфейс Movable, добавив перед его описанием служебное слово default и удалим его из всех классов, в которых он был. Такая возможность существует, начиная с Java 1.8. Теперь во всех классах будет использоваться его версия из интерфейса. В случае необходимости мы всегда можем переопределить этот метод в любом классе.

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)