Функции IDEA, которые вы не используете

Денис Гладкий
denis.gladkiy@yahoo.com, d.gladkiy@ftc.ru

Управление предоплаченных карт, ГК ЦФТ

2 января 2016 г.

1 Custom TODOs

В статье рассматриваются некоторые возможности среды разработки IntelliJ IDEA (далее, просто IDEA), не столь широко известные пользователям данного продукта компании JetBrains. Чтобы заинтриговать читателя и мотивировать его к прочтению статьи до конца, начнём с довольно неочевидной возможности IDEA — настраиваемых «меток TODO» (т.н. custom TODOs). «Метка TODO» — это текст в комментариях кода следующего вида:

```
some code before

//TODO: this thing should be done before release

some code in the middle

/* Block comments are also
supported. TODO: remove this silly comment. */

some code after
```

Далее вместо «метка TODO» будем писать просто TODO или TODOs, если имеется в виду множество таких меток. В IDEA встроен анализ кода на предмет наличия подобных артефактов:

- предупреждение при попытке зафиксировать (commit) код в системе контроля версий (vcs) при наличии TODOs в изменённых файлах (эту опцию, кстати, можно отключить в диалоге фиксации кода);
- показ списка всех TODOs с различными видами группировки (см. Рис. 1);
- выделение специальным форматированием (цвет, стиль) метки TODO и всего текста, следующего за ней на данной строке.

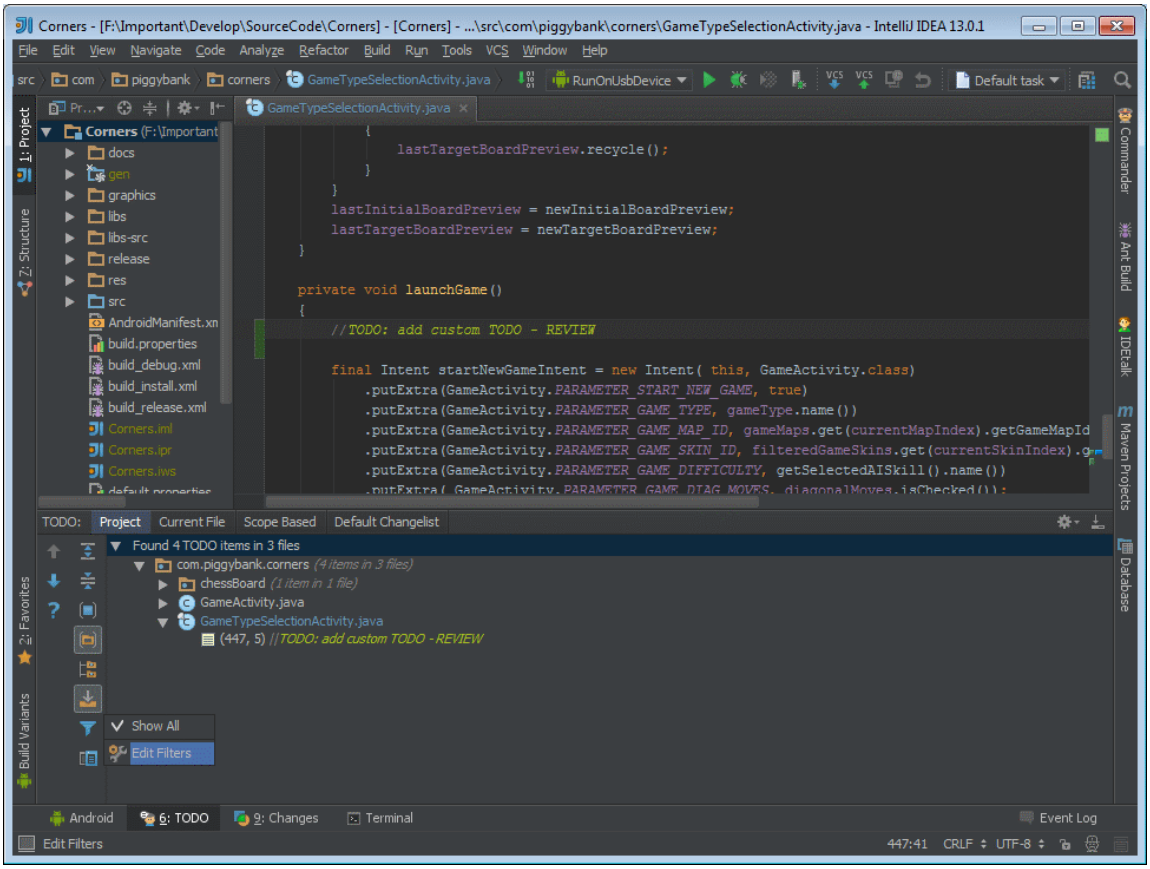

Рис. 1: Список TODO с настройками «из коробки»

IDEA позволяет конфигурировать этот наиполезнейший инструмент:

- создавать собственные шаблоны (pattern), отличные от «TODO», для текста-маркера;
- выбирать цвет и стиль выделения TODOs;
- задавать иконку для окна со списком TODOs проекта.

На рисунках 2–6 демонстрируется процесс создания нового TODO и настройки фильтра для его отображения в соответствующем списке.

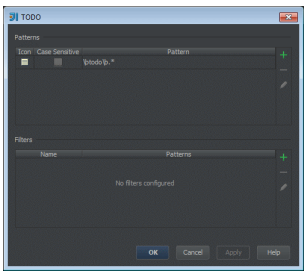

Рис. 2: Шаблоны и фильтры TODOs

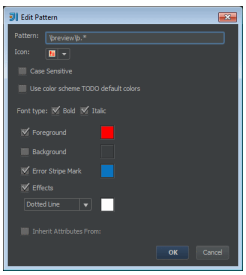

Рис. 3: Редактирование шаблона TODO

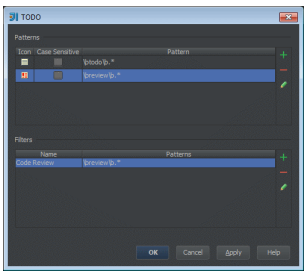

Рис. 4: Новый TODO и фильтр для него

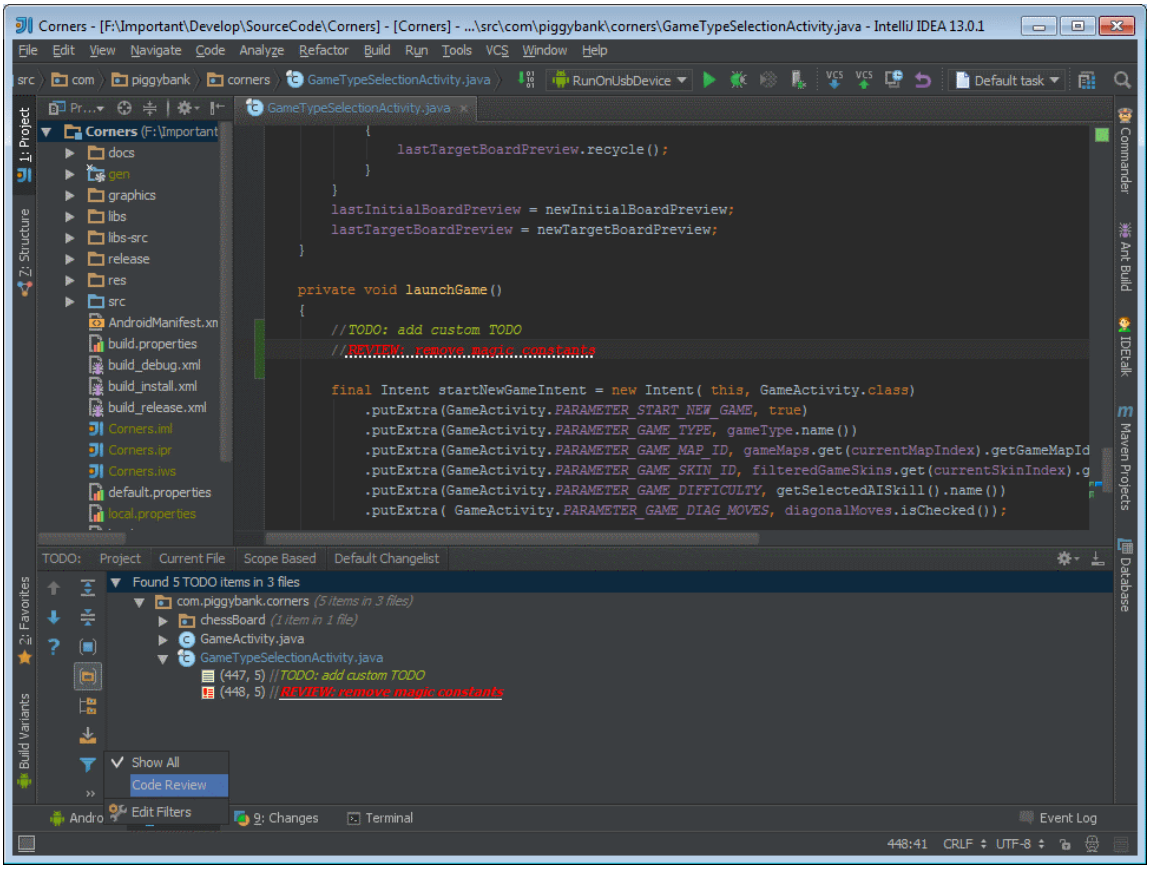

Рис. 5: Новый TODO в списке

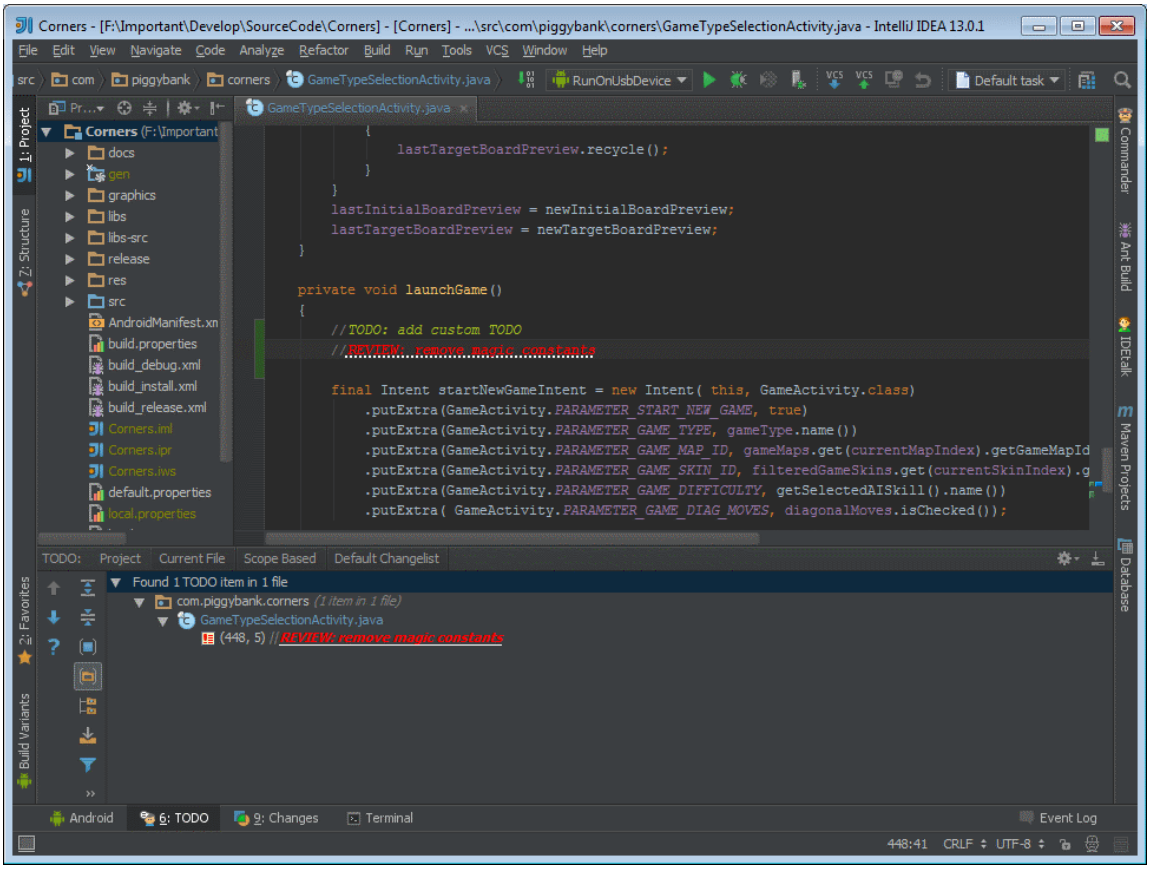

Рис. 6: Отфильтрованный список: только TODO с шаблоном «REVIEW»

Команда разработки под платформу Android использует возможность настройки TODOs для проведения аудита кода (т.н. code review). Для этого используется отдельный вид TODO: «REVIEW». Аудитор расставляет во всех подозрительных местах в коде комментарии вида «//REVIEW: some text», где «some text» — это пояснение, что и почему нужно исправить. Далее создаётся файл разницы (diff file, patch) между состояниями исходных кодов до и после аудита. Файл разницы отправляется разработчику, чей код подвергался аудиту. Разработчик применяет файл разницы на свои файлы исходных кодов (apply a patch). Результаты аудита просматриваются во вкладке TODOs, доступ к которой можно быстро получить при помощи нажатия Alt + 6 . Естественно, предполагается, что у аудитора и разработчика одинаковые настройки TODOs.

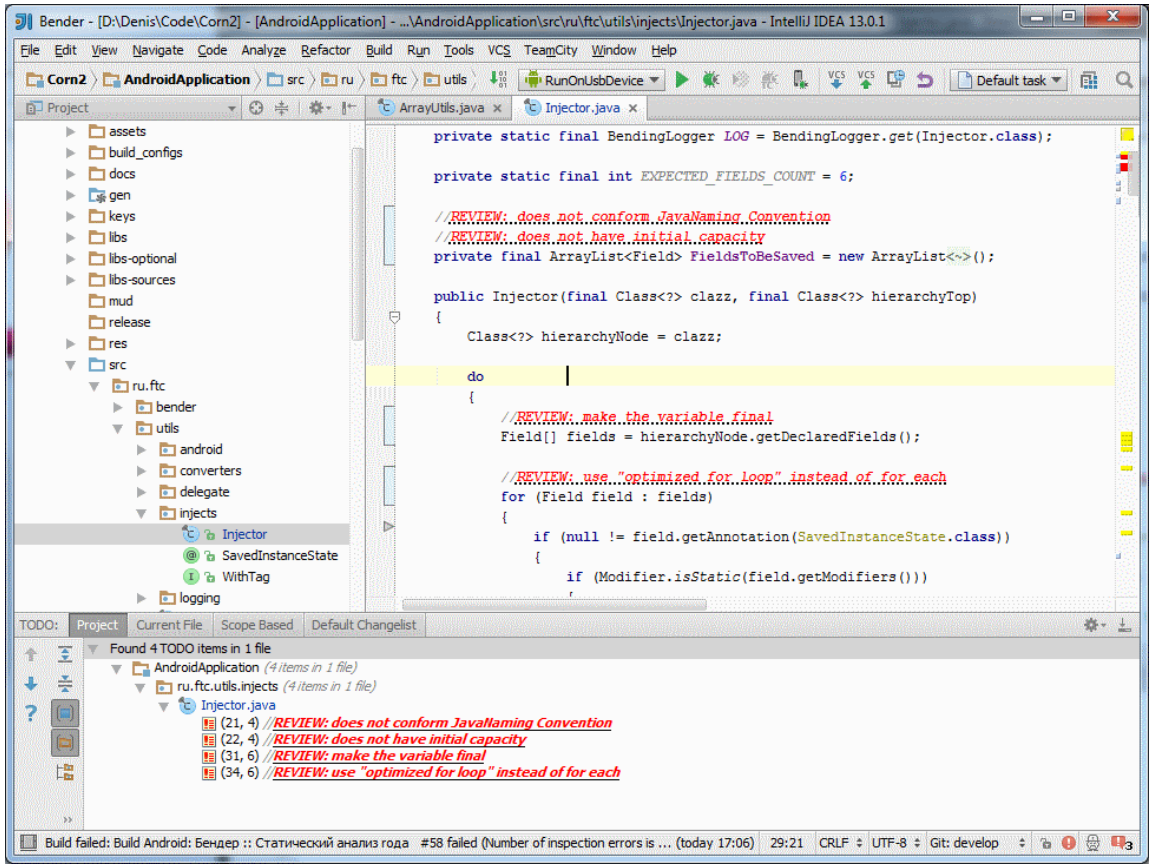

Рис. 7: Пример результата аудита кода

2 Автоматическое дополнение кода - 3 версии

Стандартная функция любой современной среды разработки — автоматическое дополнение кода (automatic code completion). Чаще всего для данной функции используется комбинация клавиш Ctrl + [ ] . В IDEA есть сразу несколько режимов работы автодополнения кода:

- Ctrl + [ ] — обычный режим авто-дополнения: подстановка имени класса, метода или переменной;
- Ctrl + Shift ⇧ + [ ] — интеллектуальный режим авто-дополнения: применяется фильтрация с учётом ожидаемого в данной точке кода типа;

- Ctrl + Shift ⇑ + Enter — авто-дополнение оператора (statement completion).

Подробное описание данных операций оставляется читателю в качестве упражнения. Хотелось бы заметить, что полезным является знание наизусть хотя бы основных горячих клавиши, описанных на странице помощи к IDEA: `http://www.jetbrains.com/idea/webhelp/keyboard-shortcuts-you-cannot-miss.html/`. [1] Интересно, что в ранних версиях IDEA была ещё одна комбинация — Ctrl + Alt + [ ], которую убрали за ненадобностью.

3 Раскраска файлов (File Colors)

Следующая функция — это раскрасска корешков вкладок файлов редактора в разные цвета. Для пояснения лучше всего показать картинку - Рис. 8.

Зачем эта функция нужна? Прежде всего, она упрощает навигацию по открытым вкладкам редактора кода, позволяя быстро ответить на вопрос: «А где я нахожусь»? Например, рассмотрим вариант использования (use case) *разработка нового UI-функционала*. В приложениях для мобильных платформ чаще всего эта задача включает работу с тремя видами функционала: непосредственно отвечающий за UI код, модель данных и сетевой код. Соответственно, для быстрого перемещения по вкладкам редактора можно их раскрасить, дабы не вчитываться в названия классов на них.

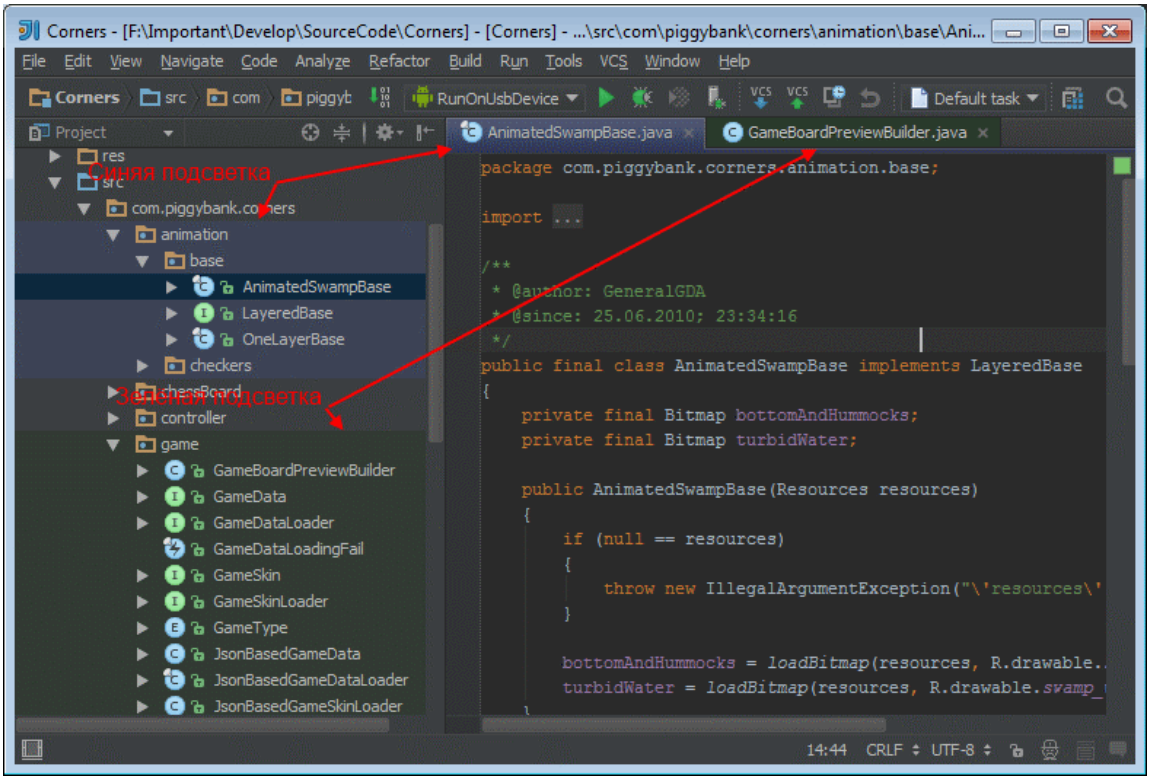

Рис. 8: Пример раскраски вкладок редактора и дерева проекта

Итак, инструкция по раскраске вкладок:

1. Открываем настройки: «File» →«Settings» (Alt + F , T).

[1] Страница по ссылке доступна на 05.01.2014

2. Выбираем пункт «File Colors» в разделе «Project Settings».

3. Жмём кнопку «+» в нужной нам секции:

   - Local colors — данные настройки раскраски будут сохранены в «локальную» часть проектного файла (*.iws в случае file-based или *.workspace в случае directory-based схемы хранения проекта IDEA). По умолчанию данная часть не фиксируется IDEA в системе контроля версий.
   - Shared colors — данные настройки раскраски будут сохранены в «общую» часть проектного файла, фиксируемую IDEA в системе контроля версий.

Проделав вышеозначенные шаги, читатель столкнётся с непродуманностью UI в данном месте. Цвет раскраски задаётся на файлы, определяемые «областью видимости» (scope), которые нельзя редактировать из меню создания новой раскраски. Для этого придётся выйти из него в меню «Scopes», расположенное в той же секции «Project Settings» (или нажав кнопку «Manage Scopes...», что всё равно не уменьшает количество кликов мышкой). Здесь можно создать требуемые области видимости — UI этого меню, думаю, будет интуитивно понятен читателю и не потребует пояснений.

4 Не нужны javadoc в форме HTML

Полезной особенностью IDEA является наличие функционала по разбору javadoc-комментариев в исходных кодах (показываются при нажатии Ctrl + Q, если каретка находится на идентификаторе, снабжённом ими). Причём, среде разработки не важно, где находятся файлы этих исходных кодов: в zip-архиве, в jar-файле или же просто в папках файловой системы. Это позволяет при включении в проект сторонних библиотек не подключать скомпилированные javadoc — большинство популярных Java-библиотек (даже классы из JDK) имеют открытый код, поставляемый в дистрибутиве.

5 Локальная история изменений

Большинство читателей наверняка уже привыкло к автосохранению файлов средой разработки. IDEA пошла в этом плане дальше: она ведёт журнал изменений в файле с возможностью сравнения и отката к конкретной версии файла. По сути, в IDEA встроена миниатюрная система контроля версий, с автоматической пометкой ревизий при различных событиях. Данный функционал доступен через меню приложения: «VCS» →«Local History» →«Show History» (Alt + S , H , H).

6 Acceleration Keys

Внимательный читатель уже наверняка заметил, что все пути по меню, описанные в статье, снабжены соответствующей комбинацией горячих клавиш. Что бы их узнать, вовсе не надо копать документацию или использовать поисковые

сервисы в интернете: *Acceleration keys* — стандартный механизм назначения горячих клавиш ещё со времён Windows 2000 (возможно, и раньше, но автору не довелось на должном уровне работать с более старыми операционными системами). В любом окне IDEA вы можете видеть, что в текстах кнопок, в названиях пунктов меню и в прочих интерактивных элементах есть подчёркнутые буквы. Эти буквы и есть горячие клавиши, которые нужно нажать в комбинации с Alt, дабы инициировать взаимодействие с элементом пользовательского интерфейса. Для примера рассмотрим снимок экрана, представленный на Рис. 9. Открытие окна «TODOs» происходит при нажатии Alt + 6. А для вызова диалога переключения между задачами нужно нажать Alt + T, T, S, Enter (последнее нажатие нужно из-за того, что в меню есть два пункта с acceleration key S).

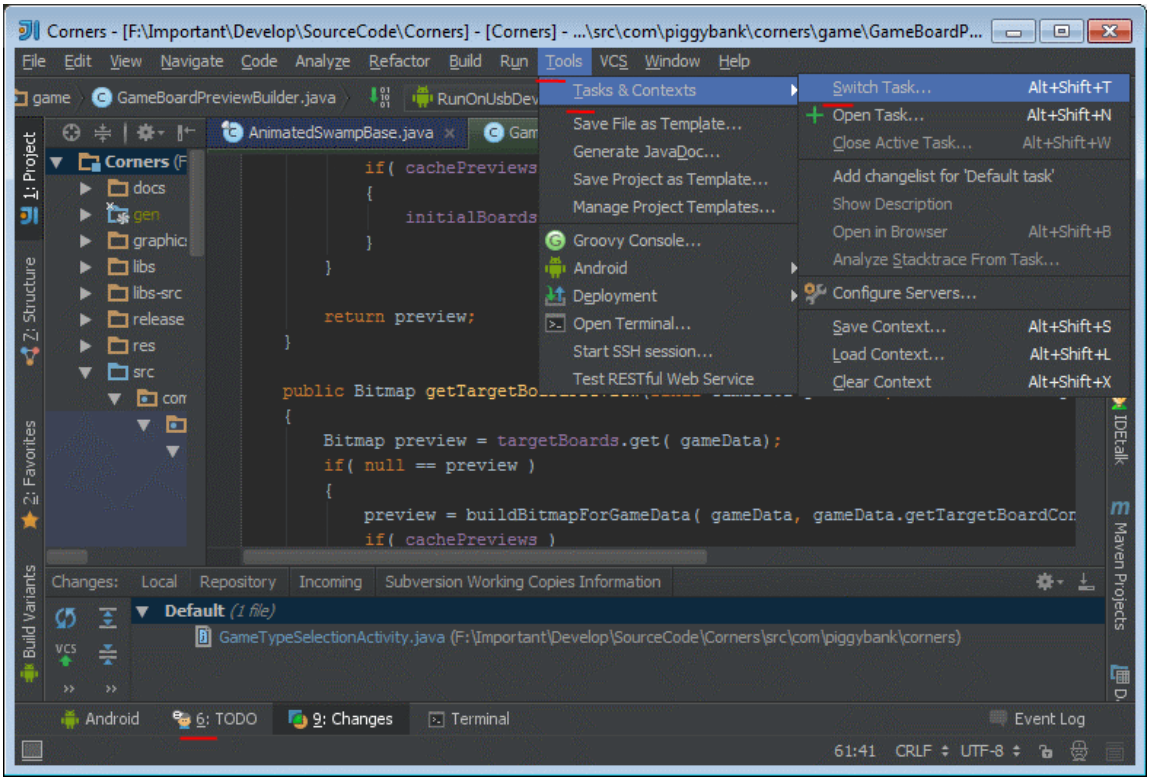

Рис. 9: Acceleration Keys

7 Управление задачами

На данный момент процесс программирования редко обходится одним лишь непосредственным написанием кода в текстовом редакторе. Современному программисту приходится иметь дело с управлением версиями кодовой базы (version control system, vcs), системой журналирования задач (bug tracking system), отчётами о потраченном рабочем времени и прочими подобными инструментами. IDEA имеет встроенный функционал по управлению «задачами». Для того, чтобы разобраться в нём, определим несколько терминов:

- контекст — открытые вкладки редактора кода + позиции курсора в них, состояние дерева проекта;

- набор изменений — именованный список изменённых файлов (в контексте системы контроля версий кодовой базы);
- запрос — конкретный запрос (issue) в системе баг-трекинга;

Теперь можно дать определение «задачи» — это *контекст*, *набор изменений* и *запрос*. IDEA даёт возможность управления задачами:

- переключение — текущий контекст меняется на контекст задачи, текущий набор изменений переключается на набор изменений задачи;
- закрытие — автоматическая фиксация изменений в системе контроля версий (комментарий генерируется по шаблону, который можно редактировать), разрешение/закрытие запроса;
- создание — очистка контекста (опционально), смена статуса запроса на «In Progress», создание нового набора изменений.

Так же есть функциональность по автоматическому измерению времени, потраченного на задачу. Поддерживается большинство современных систем баг-трекинга (JIRA, Redmine, Trac, GitHub и прочие). Естественно, самые широкие возможности IDEA предоставляет при использовании продукта от JetBrains — YouTrack. Управление задачами доступно через меню «Tools» →«Tasks & Context».

8 Снова о полезных горячих клавишах

В заключение хотелось упомянуть ещё одну пару комбинаций горячих клавиш. Первую из них автор узнал не так давно: Ctrl + Alt + N совершает переработку кода «inline variable», то есть подставляет выражение, присваеваемое переменной, в места её использования. Стоит заметить, что обратную операцию («introduce variable» через Ctrl + Alt + V) приходится использовать чаще. Вторая комбинация клавиш будет очень удобна программистам, работающим как с IDEA, так и с Visual Studio. Собственно, именно так к её использованию автор и пришёл. В IDEA перемещение по позициям редактирования осуществляется чаще всего с помощью Ctrl + Alt + ← (переход к предыдущему месту редактирования) или Ctrl + Alt + → (переход к следующему месту редактирования). В Visual Studio аналогичных команд автор не нашёл (плохо искал?) и пользовался довольно удобной альтернативой, покрывающей бóльшую часть вариантов использования — Ctrl + ⟷ и Ctrl + Shift ⇑ + ⟷ . Причём данные команды оказались настолько удобными, что после их обнаружения в IDEA использование классической навигации по местам редактирования оказалось даже менее удобным.

9 Заключение

Современные средства поддержки программирования (среды разработки, системы контроля версий и т.п.) далеко ушли вперёд от связки «блокнот+компилятор».

Они сложны сами по себе и требуют некоторых усилий по освоению их возможностей. Поэтому изучайте используемые вами инструменты, читайте списки нововведений, запоминайте наизусть горячие клавиши, и у вас появится больше времени на главную обязанность любого программиста — думать.