CodeRoid: a Framework for Generating Java Sourcecode Using Tablets and Smartphones

CONFERENCE PAPER · APRIL 2015

DOI: 10.15224/978-1-63248-044-6-27

3 AUTHORS, INCLUDING:

Fadi Wedyan

Hashemite University

13 PUBLICATIONS **58** CITATIONS

SEE PROFILE

Hani Bani-Salameh

Hashemite University

16 PUBLICATIONS **20** CITATIONS

SEE PROFILE

Proc. of the Second Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

CodeRoid: a Framework for Generating Java Sourcecode Using Tablets and Smartphones

Fadi Wedyan, Suzan Wedyan, and Hani Bani-Salameh,

Abstract—Smartphones and tablets are gaining more popularity due to their increasing capabilities and affordable prices. While most of the applications built for these devices are business, social, or entertainment oriented, few applications provide tools for developers. This is because of the limited screen size and the virtual keyboards that do not provide a comfortable programming environment. In this paper, we propose a framework, called CodeRoid, for developing Java programs that can be used on tablets and smartphones. Using CodeRoid, developers visually interact with the device to specify the specifications of the required code and CodeRoid automatically generates the required Java source files. CodeDroid provides the building programming blocks which the developer can choose and customize. CodeRoid checks the syntax of the produced Java files. In order to minimize the memory and computational requirements. Java source files can be sent to a server to produce the bytecode. Therefore, a complier and a JVM are not required. CodeRoid is developed with Android and set to run on smartphones and tablets with different screen sizes, computational and memory capabilities. The current version of the framework generates Java sourcecode, however, it can be extended to support other object-oriented programming languages (e.g., C++).

Keywords—Software tools, development environment, Java, Android, Smartphone, Tablet

I. Introduction

The rapid technological advances are changing many aspects of our lives and altering the way we work, study, or entertain. It is apparent today that the power of technology appear to be unlimited, unpredictable, and it has contributed hugely to our mobility, which has lead us from the decedent huge machines with the goal of minimization and optimization to the smartphone devices that are ubiquitous. Most of the applications on smart devices can be categorized as either business, social, educational or entertainment oriented. Smartphones and tablets are not being effectively utilized for developing due to some limitations of the devices. Mainly, the devices limited screen size, limited power, limited processors, and the limitations of the virtual keyboard, which make such devices inconvenient environment for software developing. However, these limitations are gradually oppressed. For example, manufactures began producing Smartphones with larger screen sizes and increasing computational power.

Tablets come with screen sizes that are suitable for developing. Battery lives keep increasing. These advances in the hardware technology encourage developing applications that can be used for software development. In this paper, we propose a framework, called CodeRoid, (Coding on Android) for developing Java programs that can be used on tablets and smartphones. Using CodeRoid, developers visually interact with device to specify the specifications of the required code, and CodeRoid automatically generates the required Java source files without requiring a Java compiler to be installed on the tablet or smartphone. Therefore, considering the limited power and memory resources of tablets and smartphones. The specifications can be given to CodeRoid by customizing programming building blocks. CodeRoid then generates the corresponding Java source code with limited typing requirements from the developers. The framework has the following attributes:

1) Provides a comfortable developing environment that takes into consideration the characteristics of the screen and keyboard of a tablet or a smartphone.

2) Generates code by instructions given by developers. These instructions are validated to produce a syntactically correct code.

3) Does not require installing a compiler on the device, therefore saving the limited memory and processing resources of the device.

4) CodeRoid is developed as a framework in order to allow extending it to generate code for other object-oriented programming languages.

CodeRoid is designed and implemented keeping in mind the importance of achieving the following attributes: reliability, usability, extensibility, and robustness. CodeRoid is developed with Android and set to run on devices with different screen sizes. The framework has limited memory requirements (requires about 2 MB) and works offline. We tested the framework using different devices including Samsung Galaxy S3, Samsung Galaxy S4, Samsung Note 3, and Samsung Galaxy Tab 3 (with screen sizes 7 and 8 inches). The extensibility feature of CodeRoid allows us to extend the framework to run on other platforms (e.g., iOS for iPhones) and develop other similar object-oriented programming languages (e.g., C++).

The rest of the paper is organized as follows. In Section II, we summarize related work in using smartphones and tablets for developing Java programs, and in auto-generation of Java

Fadi Wedyan and Hani Bani-Salameh
Department of Software Engineering, Hashemite University
Zarka, 13315 Jordan

Suzan Wedyan
Department of Computer Science, Amman Arab University
Amman, Jordan.

Proc. of the Second Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

programs. We describe how CodeRoid is designed and implemented in Section III. In Section IV, we demonstrate how CodeRoid can be used to develop Java programs. Finally, conclusions and future work are given in Section V.

II. RELATED WORK

Few IDE's are available for developing Java, Android, or C/C++ on smartphones and tablets. All these applications require the developer to code using an editor. Some of these applications require a Java compiler and a JVM to be installed on the device. Terminal IDE [1] is a terminal application, with a full Java/C/C++/HTML/Android development kit. It uses the command line, with many powerful and robust open-source applications, plus a custom ASCII on-screen soft keyboard that works well (You must enable it in your device's main Keyboard Settings).

AIDE [2] is an IDE for developing Android application directly on smartphones and tablets. The application allows writing code with an editor with code auto-completion, realtime error checking, refactoring, and smart code navigation. AIDE features interactive lessons with step-by-step instructions to learn Android applications development and Java programming skills. AIDE supports building applications with Java/XML and the Android SDK, applications with C/C++ and the Android NDK as well as pure Java console applications. AIDE is compatible with Eclipse projects. Developers can copy the sourcecode to their devices and open the Eclipse project in AIDE to start coding.

C4droid [3] is a C/C++ IDE for Android devices. The application requires the compiler to be installed on the device. C4droid editor has the features of syntax highlighting, tabs, code completion, code formatting, file association and undo/redo. C4droid also provides a Debugger with breakpoints and watches and a makefile tool.

Bright MIDE [4] is an integrated development environment (MIDE) for Android devices with which a developer can create, modify and build Android applications on your Android tablet or phone. Bright MIDE offers a rich text editor with syntax highlighting, auto-completion, complete file history, integrated (offline available) SDK documentation and more than 30 hot key functions of the Eclipse IDE. The editor allows quick navigation with file tabs and location history, fast and precise text selection and text operations like moving lines or commenting out of code. Bright MIDE can also work with many different projects in parallel.

Sand [5] is another IDE for Java in Android platform. Sand has a Java editor with keywords highlighting for Java, showing line number, undo/redo feature, and automatically close are all supported. Sand has a full-featured Java compiler and can run Java programs. Output and input are both supported by the console of Sand.

Existing research on auto-generation of Java code concentrates on generating Java class templates or bytecode from various UML diagrams. This includes generating Java executables from class diagrams and state diagrams (e.g., [6]–[8]), approaches for generating Java executables from class diagrams, sequence diagrams, and activity diagrams (e.g., [9], [10]), or approaches that produces Java code from formal specifications (e.g., [11]). However, all these approaches are designed to run on computers and are partially implemented. While generating Java code (whether sourcecode or bytecode) have many potentials including cost reduction and accuracy, the road is still long to reach a system that can be reliable and efficient.

III. APPLICATION DESIGN AND IMPLEMENTATION

We followed the Incremental Methodology in developing CodeRoid [12]. We choose this approach due to it's features that are suitable to our application. Mainly, it ggenerates a working software early during the software life cycle, easier to test and debug during a smaller iteration, and easier to manage risks [12]. Figure 1 shows the package diagram for CodeRoid. The framework consists of the following components:

- Codroid Sketcher. A Singleton that starts the main activity of the framework.
- Validation. The component provides the necessary classes for validating the input data.
- Sketcher Exception: The component provides exception classes for handling different types of runtime errors.
- GUI. Contains the classes necessary for creating the GUI.
- Entities. the package that works as the structure of the files which is workspace, projects and packages.

Our design and implementation of CodeRoid achieves the following attributes:

- Reliability. Control flows in CodeRoid as consistent wizard that give developers the ability to perform their work with minimum failures. A wizard is a user interface type that presents a user with a sequence of dialog boxes that lead the user through a series of well-defined steps to easily perform complex tasks. When a failure occurs, CodeRoid recovers quickly and resumes work from the failure point.
- Usability. CodeRoid is a well structured wizard which is easy to learn and master. Using the framework does not require any classroom training. CodeRoid has a consistent engaging interfaces to explore the project, packages and files. In the same way, developers can easily generate and modify the code just by pressing a button. CodeRoid is provided with informative help messages to further increase usability.

Proc. of the Second Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

- Extensibility. CodeRoid is built as a framework that allows adding plug-ins for various future requirements and functionalities.
- Robustness. CodeRoid generates code following instructions given by the developers; these instructions protect the developers from making a syntactic mistake in the programming language syntax.

In the designing of the Graphical User Interface (GUI), we followed the following design principles:

1) Strive for cconsistency. The design of all the interface components used in prompts, menus, and help screens; and consistent commands had been employed throughout the framework.

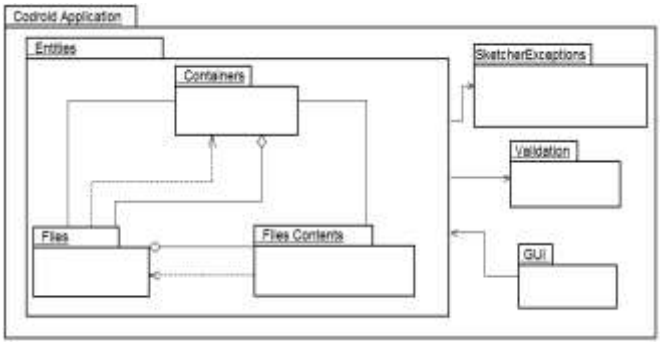

Figure 1. Package Diagram of CodeRoid

2) Enable the use of Short Cuts. We reduced the number of interactions to increase the pace of interaction. Abbreviations functioning keys and hidden commands are very helpful even to an expert user.

3) Offer informative feedback. Every action performed by the user is handled with system feedback. For frequent actions, the response can be modest. For infrequent actions, the response is handled more substantially. For example, when the developer creates a class the response would be shown as the last modification in the Code Logger, since creating a class is a frequent action. When the developer types an invalid name, which is an infrequent action, an informative notification message is displayed.

4) Offer ssimple eerror handling. CodeRoid is designed in away that minimizes developers incorrect usage. However, incorrect interactions cannot be fully eliminated. For these situations, the framework detects the errors and offers simple handling for the errors in a comprehensible mechanisms.

5) Permit easy reversal of actions. developers can simply undo or cancel any action before committing to this action.

The following tools were used to implement CodeRoid:

• Microsoft Visio 2013 [13], for preparing the design documents. We used UML [14] to describe the system architecture.

• Eclipse Indigo [15] with android plug-in.

• Adobe Photoshop CS5 [16], for designing icons and images.

IV. DEMONSTRATION

In this section, we demonstrate how CodeRoid can be used to develop Java programs. Figure 2 shows the first screen displayed when a developer starts CodeRoid. A developer can choose either between opening an existing project and creating a new one. Figure 3 shows the screen displayed by CodeRoid when the developer chooses to create a new project. The developer can choose which package (i.e., subsystem) he/she want to create. CodeRoid allows a developer to create a software incrementally by having the option of creating subsystem gradually. Developers can choose names according to Java naming conventions.

Figure 2. Welcome Window of CodeRoid

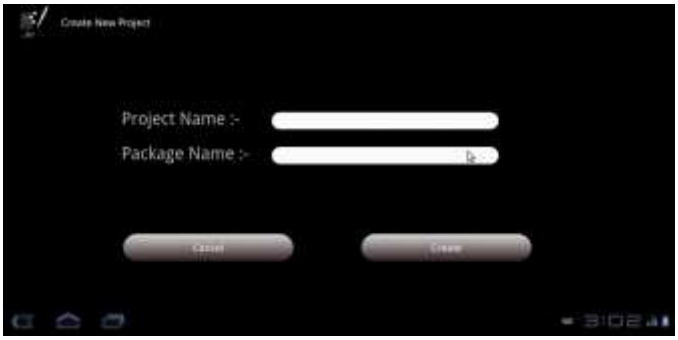

Figure 3. Creation of Projects and Subsystems

Figure 4 shows the main screen of CodeRoid. The panel on the right provides tools such as edit, save, delete, or refresh for a chosen component. In the bottom of the window, the developer can pick up one of the available components. A developer can choose to define one of the structure components (i.e., A class, an interface, or a package), or to define a variable with a type. A developer can choose from primitive data types, or class types available in the Java library. A developer can also import class libraries when needed for the project which will be added to the available types in the bottom panel. CodeRoid allows developers to nest

Proc. of the Second Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

package (i.e., defining a package inside another package). In Figure 5, we show the window that gets displayed when the developer chooses to create a new package.

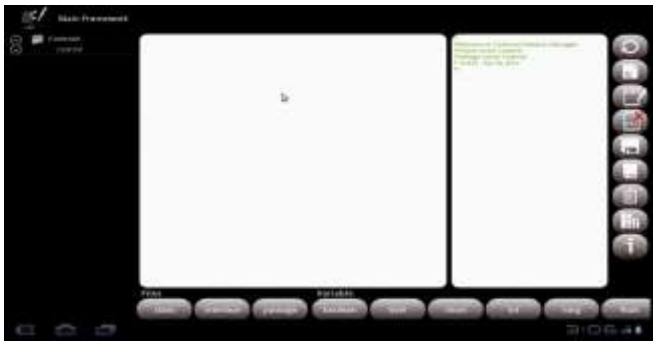

Figure 4. CodeRoid Main Screen

Figure 5. Create New Package Window

In Figure 6, we show how a developer can create a class using CodeRoid. Note that the developer only need to type-in one field, which is the field used for entering the class name. For other options (accessibility, modifiers), the developer only needs to select the required option. Developers can also specify the superclass using a drop-down list, and in which subsystem the class should be created. The list in the bottom of the window shows the interfaces the class implements, which can specified by clicking the drop-down list above it. Similar to the creation of classes, a developer can create an interface as shown in Figure 7. The developer types the interface name, and specifies the package where the interface belongs and the interfaces implemented by the new created interface. If the developed chooses a name for the interface similar to an existing one, an error message gets displayed to indicate that .

After a class or interface is created, a developer can start adding components to each of them. Figure 8 shows how a developer can add a method to a class. The developer types in the method name and selects the required method modifiers and accessibility options. The developer proceeds by choosing the method return type and parameters. To add a parameter, the developer types-in the parameter name and selects the type from a drop-down list. A list in the bottom of the window shows the methods parameter added so far. The developer can chose a parameter from the list and delete it (if needed).

Figure 6. Create New Class Window

Figure 7. Create New Interface Window

CodeRoid also allows the developer to change the order of the parameters by moving a parameter up and down in the list. Constructors can be created similar to methods as shown in Figure 9 but without the return type option and without the final and static modifiers (which are not allowed for constructors in Java). In Figure 10, we show how developers can define and add instance variables to a class using CodeRoid. The developers can specify the accessibility options and modifiers, the instance variable type, and the variable name. Note that the drop-down list for the types contain the primitive data types, types defined in java.lang, and any user defined types specified in the project classpath.

Proc. of the Second Intl. Conf. on Advances in Information Processing and Communication Technology - IPCT 2015

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

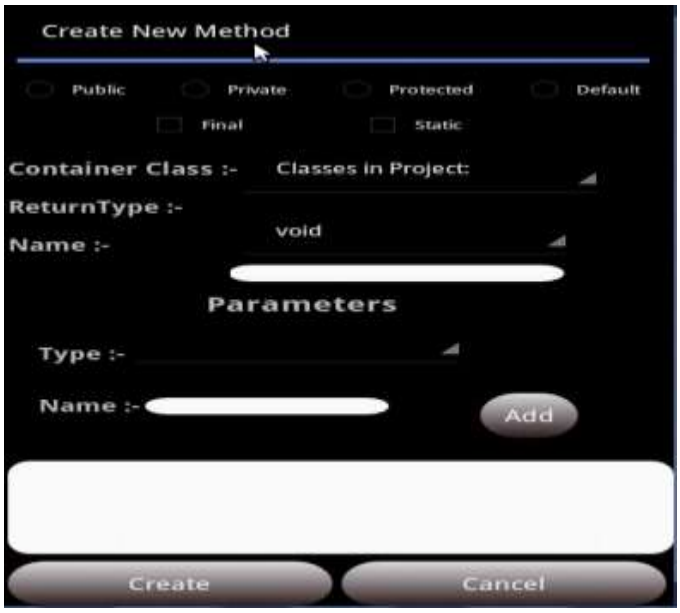

Figure 8. Create New Method Window

v. CONCLUSIONS AND FUTURE WORK

The revolution of mobile devices opened perspectives for different types of application. In this paper, we presented an extensible and maintainable framework that allows developers to use their smartphones or tablets to develop Java programs. We developed CodeRoid with Android and we set it to run on devices with different screen sizes. The framework currently generates Java source code with minimum typing required from the developer. We designed CodeRoid keeping in mind the importance of having a framework that is reliable, usable, extensible, and robust. These features are achieved by following the design and implementation principles of OOP. We consider CodeRoid as the first step toward developing a visual source code generator for programming languages that can be used on smart devices. The presented work can be further extended in many directions including:

1) Including classes from Java API (e.g. classes List, Set, ArrayList from collections library).
2) Including constructs for annotations.
3) Adding the option of compiling the generated Java source files using a website. The generated bytecode can be returned back to the device or stored on the site.
4) Developing versions of the framework that work on different platforms (e.g., iOS for iPhones).
5) Extending the framework for other object-oriented programming languages (e.g., C++).

Figure 9. Create New Constructor Window

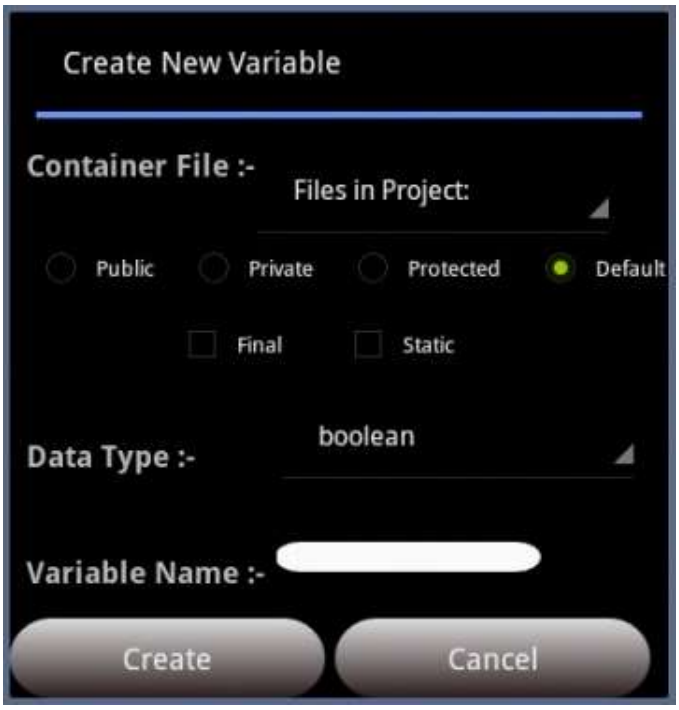

Figure 10. Create New Variable Window

ISBN: 978-1-63248-044-6 doi: 10.15224/ 978-1-63248-044-6-27

Acknowledgment

This work was supported in part by the Hashemite University under grant #2013/19 to F. Wedyan and H. Bani Salameh.

References

[1] Spartacus Rex, Terminal IDE, https://play.google.com/store/apps/details?id=com.spartacusrex.spartacuside&hl=en, 2014.

[2] appfour, AIDE - Android IDE - Java, C++, https://play.google.com/store/apps/details?id=com.aide.ui&hl=en, 2014.

[3] C4droid - C/C++ compiler and IDE, https://play.google.com/store/apps/details?id=com.n0n3m4.droidc&hl=en, 2014.

[4] Philip Heyse, Bright MIDE: Java/Android IDE, https://play.google.com/store/apps/details?id=de.bright side.brightmidemain&hl=en, 2014.

[5] Sand IDE Pro for Java, https://play.google.com/store/apps/details?id=com.jimmychen.app.sand.pro&hl=en, 2014.

[6] I. A. Niaz, J. Tanaka et al., "Mapping uml statecharts to java code." in IASTED Conf. on Software Engineering, 2004, pp. 111–116.

[7] R. Pilitowski and A. Derezinska, "Code generation and execution ˜ framework for uml 2.0 classes and state machines," in Innovations and Advanced Techniques in Computer and Information Sciences and Engineering. Springer, 2007, pp. 421–427.

[8] R. Tiella, A. Villafiorita, and S. Tomasi, "Fsmc+, a tool for the generation of java code from statecharts," in Proceedings of the 5th international symposium on Principles and practice of programming in Java. ACM, 2007, pp. 93–102.

[9] M. Usman and A. Nadeem, "Automatic generation of java code from uml diagrams using ujector," International Journal of Software Engineering and Its Applications, vol. 3, no. 2, pp. 21–37, 2009.

[10] A. G. Parada, E. Siegert, and L. B. de Brisolara, "Generating java code from uml class and sequence diagrams," in Computing System Engineering (SBESC), 2011 Brazilian Symposium on. IEEE, 2011, pp. 99–101.

[11] S. Ramkarthik and C. Zhang, "Generating java skeletal code with design contracts from specifications in a subset of object z," in Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on. IEEE, 2006, pp. 405–411.

[12] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," Computer, vol. 36, no. 6, pp. 47–56, 2003.

[13] Microsoft Visio, http://office.microsoft.com/en-001/visio/, 2013.

[14] Object Management Group, "Unified modeling language," http://www.uml.org/, 2014.

[15] Eclipse Indigo, https://eclipse.org/, 2014.

[16] Adobe Photoshop CS5, http://www.adobe.com/products/photoshop.html, 2014.