[1.]Anca-Elena IORDAN

DESIGN AND IMPLEMENTATION OF A GAME USING ANDROID API

[1.] University Politehnica Timisoara, Faculty of Engineering Hunedoara, ROMANIA

Abstract: In this article are rendered the requisite stages for the accomplishment of a game. The design of the game is accomplished by the next two unified modelling language diagrams: use-case diagram and class diagram. By achieving these types of diagrams, the game is described in an obvious and concrete approach, without ambiguousness. There have been identified seven specific concepts of this game, and then there have been implemented corresponding classes for these concepts. For the game development on the Android platform it will be used computer science branches as object oriented programming and computational geometry, and as programming language it will be used Java Standard Edition with Android application programming interface. The CASE tool used to represent the diagrams was ArgoUML, and the source code was written in Eclipse integrated development environment.

Keywords: Game, UML, ArgoUML, Java SE, Android API, Eclipse IDE

INTRODUCTION

From the perspective of UML modelling language, the analysis of the software consists in the making of use-case diagram [1]. The games' options will be described in a clear manner by representing the use-cases. Each use-case presents the interactions between user (player) and the software (game). The representation of the use-case diagram is showed in figure 1.

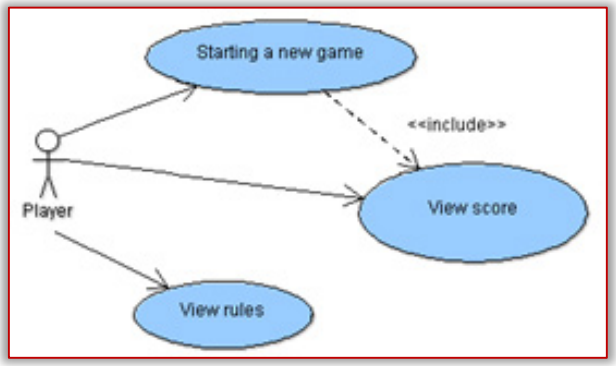

Figure 1. Use-case diagram

This diagram defines the software domain, allowing the visualisation of the dimension and the action sphere of the whole development process [2]. The diagram's structure contains:

- One actor that represents the user (player), this being the external entity which the software interacts with.
- Three use-cases which describe the functionality of the game.
- Relationships between user (player) and use-cases (association relations) and, also relations between use-cases (dependency relations).

THE DESIGN PHASE

The conceptual modelling allows identification of the most important concepts used in the game's implementation. There have been identified seven specific concepts of this game, and then there have been implemented corresponding classes [3] for these concepts. These classes are the following: "Peste", "Foc", "Animatie", "VizualizareJoc", "Joc", "Reguli" and "Principal".

The "Peste" class contains ten atributes: nine of "int" type and one atribute of "Bitmap" type defined in the *android.graphics* package [4]. Also this class has in its componence five methods. The structure of this class is showed in figure 2. It can be observed that an object of this class is made from a "Bitmap" object (composion relation), but it can be formed from a "Canvas" object defined in *android.graphics* package, two "Rect" objects defined in *android.graphics* package and a "Random" object defined in *java.util* [5] (aggregation relations).

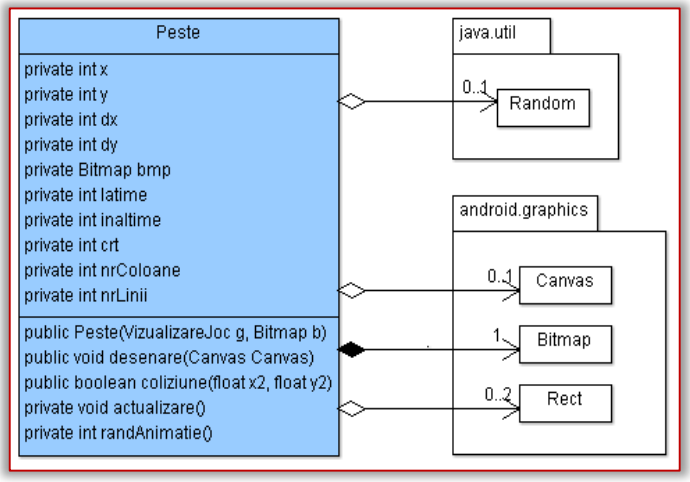

Figure 2. Structure of the "Peste" class

The "Foc" class has among its componence three atributes: two of "float" type and one atribute of "Bitmap" type defined in the *android.graphics* package. Also this class contains two methods. The representation of this class is showed in figure 3. It can be noted that an object of this class is composed of a "Bitmap" object (composition relation), but can be formed from a "Canvas" object defined in *android.graphics* package (aggregation relations).

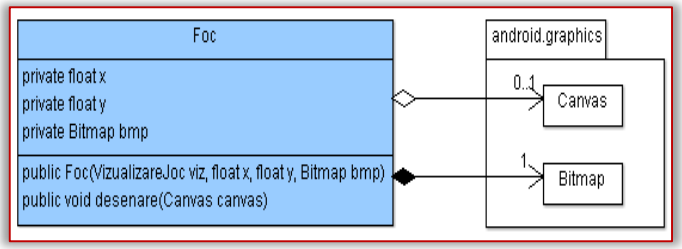

Figure 3. Structure of the "Foc" class

The "Animatie" class, which inherits the attributes and methods of the "Thread" class defined in the *java.lang* [6] package, has in its composition one attribute of "long" type and two methods. The structure of this class is showed in figure 4. It can be observed from the previous mentioned figure that an object of this class is formed from a "Canvas" object, a "Toast" object defined in the *android.widget* [7] package and an "Intent" object defined in *android.content* package. All of these three relations are aggregation relations.

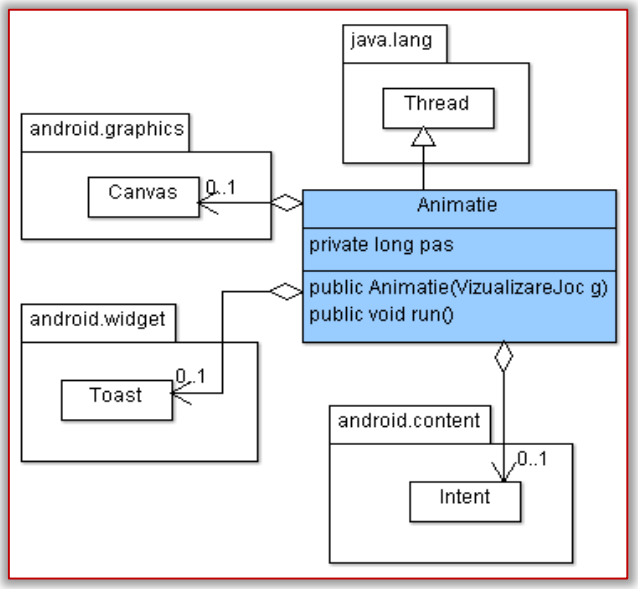

Figure 4. Structure of the "Animatie" class

The "VizualizareJoc" class, which inherits the "SurfaceView" class defined in the *android.view* package, has in its composition ten attributes: three attributes of "Bitmap" type, one attribute of "SurfaceHolder" type, two attributes of "List<Peste>" type, one attribute of "Foc" type, one attribute of "Animatie" type , one attribute of "long" type and one attribute of "int" type. The sequence from the class diagram which presents the structure of "VizualizareJoc" class is showed in figure 5. Due to the presence of the composition relations, it can be observed that an object of this class is composed of: three "Bitmap" objects, one "SurfaceHolder" object which corresponding class is defined in *android.view* [8] package, two object of the generic type "List" defined in *java.util* package, one "Animatie" object and one "Foc" object. Because of the aggregation relations existing in the class diagram, it can be observed that an object of this class is formed also from: one "MotionEvent" object defined in *android.view* package, one "Context" object defined in *android.content* [9] package, one "Random" object defined in *java.util* [10] package and one "Canvas" object defined in *android.graphics* package.

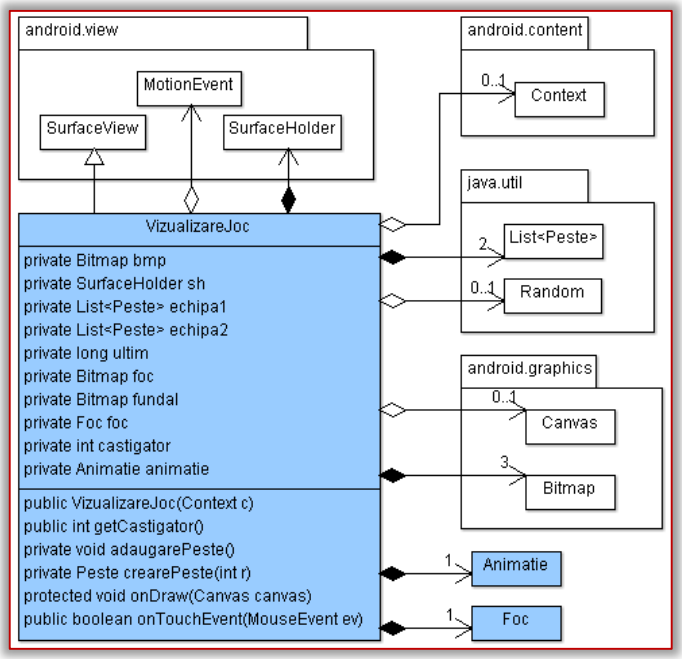

Figure 5. Structure of the "VizualizareJoc" class

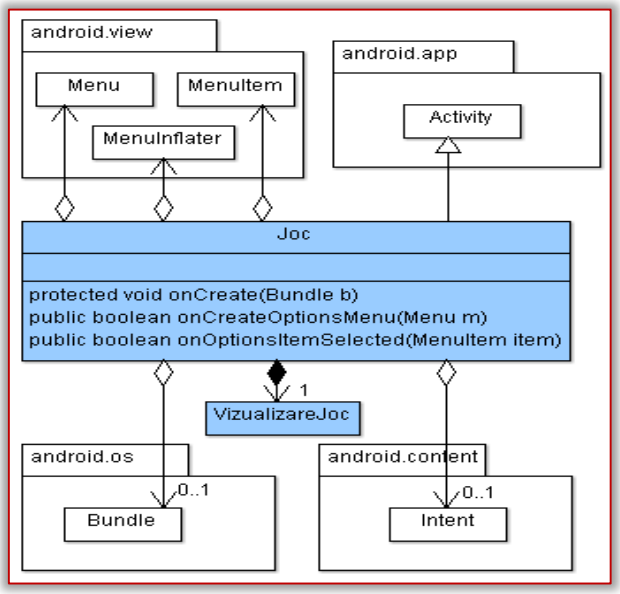

Figure 6. Structure of the "Joc" class

The "Joc" class, which inherits the attributes and methods of "Activity" class, defined in the *android.app* [11] package, has in its composition three methods. The structure of this class is presented in figure 6. Because of the presence of the composition relation, it can be noticed that an object of this class is composed from an object of "VizualizareJoc" type. Due to the presence of the aggregation relations, it can be observed that an object of "VizualizareJoc" type can be formed of an object of "Menu" type, an object of "MenuItem" type, an object of "MenuInflater" type, an object of tip "Bundle" type and an object of tip "Intent" type. The first three objects are defined in the *android.view* package, the forth object is defined in the *android.os* [12] package and the last (fifth) object is defined in the *android.content* package.

The "Principal" class, which inherits the attributes and methods of "Activity" class, has in its composition three attributes and three methods. The structure of this class is presented in figure 7. In this figure are represented the two existing composition relations, which means that an object of the "Principal" class is composed from an object of "Intent" type and two objects of "Button" type defined in the *android.widget* package. Also in the same figure are represented four aggregation relations. Their meaning consist in the fact that an object of the "Principal" class can contain: an object of "Menu" type, an object of "MenuItem" type, an object of "MenuInflater" type and an object of tip "Bundle" type.

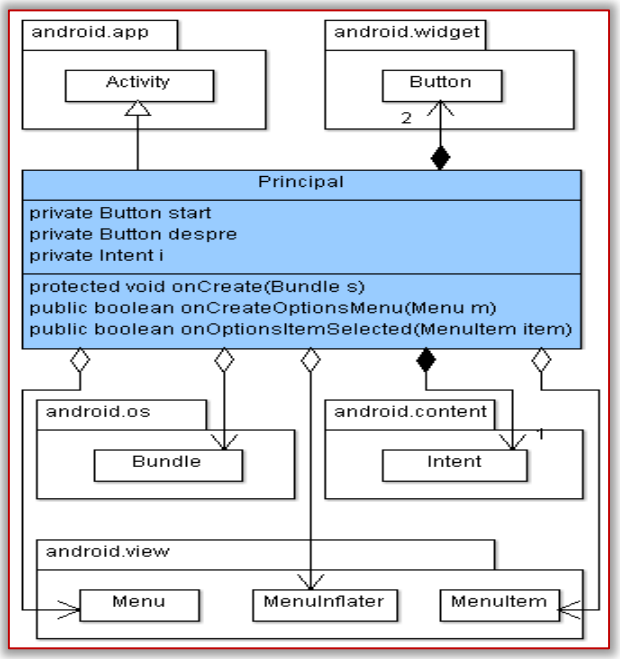

Figure 7. Structure of the "Principal" class

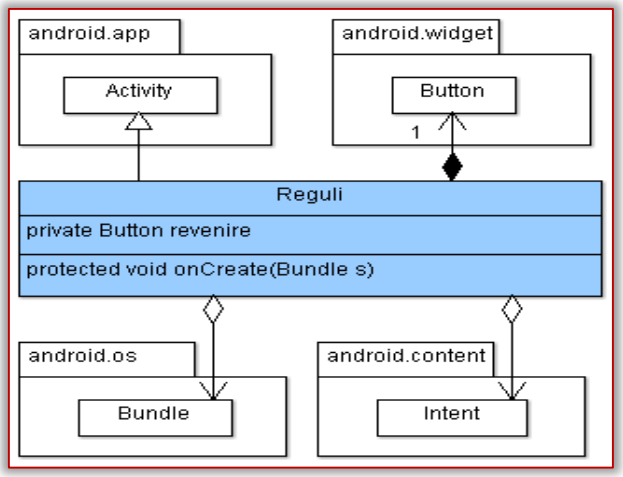

Figure 8. Structure of the "Reguli" class

The "Reguli" class, which inherits the attributes and the methods of the "Activity" class has in its composition one attribute of "Button" type and one method and it is represented in the figure 8. In this figure it can be observed that an object of this class is composed from an object of "Button" type due to the presence of the composition relation [13]. Because of the presence of the two aggregation relations [14], it can be noticed that an object of "Reguli" type can be formed from a "Bundle" type object and an "Intent" type object.

GRAPHICAL USER INTERFACE

The main user graphical interface of the game is an object of „Principal" type [15], having the graphical representation which is showed in figure 9.

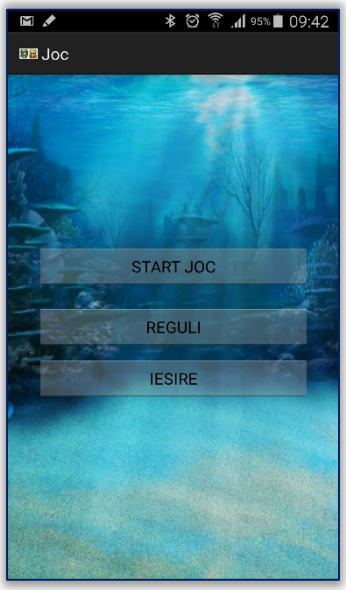

Figure 9. Main user graphical interface

By selecting the first option from the presented graphical interface, it is instantiated an object of "Joc" class, which allows the start of a new game (figure 9). To represent the two species of fish it is used an instance of "VizualizareJoc" class and to simulate the fish movement it is used an instance of "Animatie" class.

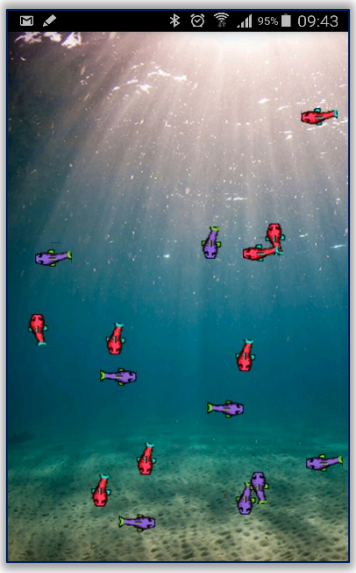

Figure 10. Graphical interface of the game

CONCLUSIONS

In this paper it was presented the development of a game based on UML design, Android API implementation, ArgoUML CASE tool and Eclipse interactive environment.

The Android SDK is a strong and full-featured set of APIs that Java developers will find usual and simple to utilize. Android

is a ripe, yet still growing, platform that many game developers have enfolded. Not only is Android a well-fixed platform, but the features of actual Android devices are starting to exceed the features of just those of latter generation console systems. The variety of users does Android an important platform for game development companies as respects the tools and services available.

Because the representation of the UML diagrams corresponding to all three phases: analysis, design and implementation, the puzzle game has been described in an obvious and objective manner, without ambiguity. The use of the unified modelling language for the achievement of the game is characterized by rigorous syntactic, rich semantic and visual modelling support.

References

[1] J. Hunt, Guide to the Unified Process featuring UML, Java and Design Patterns, Springer London Ltd, 2014
[2] B. Bruegge, A. Dutoit, Object Oriented Software Engineering Using UML, Patterns, and Java, Pearson Education, 2013
[3] A. Dennis, B. H. Wixom, D. Tegarden, Systems Analysis and Design with UML, John Wiley & Sons Ltd, 2012
[4] B. Hardy, C. Stewart, Android Programming, Pearson Education, 2015
[5] U. Roy, Advanced Java Programming, OUP India, 2015
[6] D. Vohra, Beginning Java Programming, John Wiley & Sons Ltd, 2015
[7] Z. Mednieks, G. Blake-Meike, M. Nakamura, Programming Android, O'Reilly Media Inc., 2016
[8] E. Hellman, Expert Android Programming, John Wiley & Sons Ltd, 2016
[9] B. Hardy, C. Stewart, Android Programming, Pearson Education, 2015
[10] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, The Java Language Specification, Oracle, 2013
[11] W. M. Lee, Beginning Android Programming with Android Studio, John Wiley & Sons Ltd, 2013
[12] R. Rogers, Learning Android Game Programming, Pearson Education (US), 2011
[13] F. White, Object-Oriented Software Engineering: Practical Software Development using UML and Java, McGraw-Hill Education, 2015
[14] B. Rumpe, Modeling with UML, Springer International Publishing AG, 2016
[15] B. Burd, Java Programming for Android Developers, John Wiley & Sons Ltd, 2013

ISSN: 2067-3809