Projektovanje softvera

Verzija <1.0>

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 21/12/2016 | 1.0 | Početna verzija dokumenta | Zoran Tošić |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Uvod 4

2. Definitions 4

2.1 Objektno orjentisano progamiranje u Javi: 4

2.1.1 Abstract class 4

2.1.2 Abstract method 4

2.1.3 Array 4

2.1.4 Attribute 4

2.1.5 Catch clause 4

2.1.6 Catching exceptions 4

2.1.7 Class 4

2.1.8 Class body 4

2.1.9 Class constant 4

2.1.10 Class inheritance 5

2.1.11 File system 5

2.1.12 Garbage collector 5

2.1.13 Interface inheritance 5

2.1.14 Method 5

2.2 UML 5

2.2.1 Abstract class 5

2.2.2 Action 5

2.2.3 Action state 5

2.2.4 Activity diagram 5

2.2.5 Actor 5

2.2.6 Aggregation 5

2.2.7 Association 5

2.2.8 Cardinality 5

2.2.9 Class diagram 5

2.2.10 Composite state 6

2.2.11 Composition 6

2.2.12 Deployment diagram 6

2.2.13 Diagram 6

2.2.14 Extends 6

2.2.15 Generalization 6

2.2.16 Inheritance 6

2.2.17 Metaclass 6

2.2.18 Object diagram 6

2.2.19 Sequence diagram 6

2.2.20 Use case 6

2.2.21 Use case diagram 6

2.2.22 View 6

# Uvod

Rječnik sadrži sve termine i riječi koje korisnik treba da zna da bi mogao razumjeti čitavu dokumentaciju projekta. Ovaj dokument se koristi da definiše terminologiju specifičnu domenu programa, da objasni termine koji mogu biti nepoznati čitaocu use case opisa ili drugih projektnih dokumenata.

## Svrha

Glossary je namijenjen:

- developerima – koji će koristiti termine definisane u ovom dokumentu prilikom dizajniranja i implementiranja klasa, tabela u bazi podataka, korisnički interfejs i tako dalje.

-analitičarima – koji će koristiti ovaj dokument da precizno definišu pravila poslovanja, i osiguraju da se čitav projekat realizuje uz korištenje propisanih termina.

- edukatorima i kreatorima dokumentacije – koji će koristiti ovaj document da naprave tutorijale za obuku korisnika i pisanje dokumentacije koristeći poznatu terminologiju.

**1.2 Opseg**

Opseg glossary je ograničen na trajanje datog projekta. Neki njegovi dijelovi se mogu koristiti kao referenca u drugim projektima, ali kao cjelina ne bi bio odgovarajući.

**1.3 Pregled**

Glossary je dat u obliku rječnika, gdje su termini navedeni po abecednom redu, a ispod svakog termina nalazi se objašnjenje tog termina. Kakos u termini preuzeti iz zvaničnih dokumenata, prevođenje termina i njihovih opisa narušilo bi njihov integritet, tako da je ostavljen engleski jezik.

# Definicije

## Objektno-orjentisano programiranje u Javi:

Ovde su navedeni svi termini koji se javljaju prilikom rada sa Java programskim jezikom. Termini su preuzeti iz knjige Object-Oriented Programming with Java od David J. Barnes.

### Abstract class

A class with the abstract reserved word in its header. Abstract classes are distinguished by the fact that you may not directly construct objects from them using the *new­* operetaro. An abstract *class* may have zero or more *abstract methods.*

### Abstract method

A method with the *abstract* reserved word in its header. An abstract method has no *method body.* Methods defined in an *interface* are always abstract. The body of an abstract method must be defined in a *sub class* of an *abstract class,* or the body of a class implementing an interface.

### Array

A fixed-size object that can hold zero or more items of the array’s declared type.

### Attribute

A particular usage of an *instance variable. ­*The set of attribute values held in particular *instance*  of a class define the current *state* of that instance. A class definition may impose particular constraints on the valid states of its instances by requiring that a particular attribute, or set of attributes, do not take on particular values.

### Catch clause

The part of a *try statement* responsible for handling a caught *exception.*

### Catching exceptions

Exceptions are caught within the *catch clause* of a *try statement.* Catching an exception gives the program an opportunity to recover from the problem or attempt a repair for whatever caused it.

### Class

A programming language concept that allows data and *methods* to be grouped together. The class concept is fundamental to the notion of an *object-oriented ­*­*programing language.* The methods of a class define the set of permitted operations on the class’s data.

### Class body

The body of a *class* definition. The body groups the definitions of a class’s *members – fields, methods* and *nested classes.*

### Class constant

A variable defined as both *final* and *static.*

### Class inheritance

When a *supper class* is extended by a *sub class,* a class inheritance relationship exists between them. The sub class inherits the methods and attributes of its supper class. In Java, class inheritance is *single* inheritance*.*

### File system

An *operating system* makes it possible to use space on a computer’s *disk drives* by imposing a structured file system on the disk storage. Each file system has its own conventions for the way in which files are named, folders and directories are structured, and large files are split into smaller pieces, for instance.

### Garbage collector

A *daemon thread* that recycles objects to which there are no extant references within a program.

### Interface inheritance

When a *class* implements an *interface,* an interface *inheritance* relationship exists between them. The class inherits no implementation from the interface, only method signatures and *static variables.* It is also possible for one interface to extend one or more interfaces. In Java, interface inheritance is the only form of *multiple inheritance.*

### Method

The part of a *class definition* that implements some of the behavior of objects of the class. The body of the method contains *declarations* of *local variables* and statements to implement the behavior. A method receives input via arguments, if any, and may return a result if it has not been declared as *void.*

## UML

### Abstract class

A class that cannot be directly instantiated.

### Action

A computational or algorithmic procedure.

### Action state

A state with an internal action and one or more outgoing transitions involving the completion of an internal action.

### Activity diagram

A special case of a state diagram in which all or most of the states are action states and in which all or most of the transitions are triggered by completion of actions in the source states.

### Actor

A predefined stereotype of type denoting an entity outside the system that interacts with use cases.

### Aggregation

A special form of association that specifies a whole-part relationship between the *aggregate* (whole) and a component part.

### Association

A relationship that describes a set of links.

### Cardinality

The number of elements in a set.

### Class diagram

A diagram that shows a collection of declarative model elements, such as classes, types, and their contents and relationships.

### Composite state

A state that consists of substates.

### Composition

A form of aggregation with strong ownership and coincident lifetime as part of the whole. Parts with non-fixed multiplicity may be created after the composite itself, but once created they live and die with it.

### Deployment diagram

A diagram that shows the configuration of run-time processing nodes and the components, processes, and objects that live on them.

### Diagram

A graphical presentation of a collection of model elements, most often rendered as a connected graph of arcs and vertices. The UML supports the following diagrams: *class diagram, object diagram, use case diagram, sequence diagram, collaboration diagram, state diagram, activity diagram, component diagram, deployment diagram.*

### Extends

A relationship from one use case to another, specifying how the behavior defined for the first use case can be inserted into the behavior defined for the second use case.

### Generalization

A taxonomic relationship between a more general element and a more specific element. The more specific element is fully consistent with the more general element and contains additional information. An instance of the more specific element may be used where the more general element is allowed.

### Inheritance

The mechanism by which more specific elements incorporate structure and behavior of more general elements related by behavior.

### Metaclass

A class whose instances are classes.

### Object diagram

A diagram that encompasses objects and their relationships at a point in time. An object diagram may be considered a special case of a class diagram or a collaboration diagram.

### Sequence diagram

A digram that shows object interactions arranged in time sequence. In particular, it shows the objects participating in the interaction and the sequence of messages exchanged.

### Use case

A class that defines a set of *use case instances.*

### Use case diagram

A digram that shows relationsips among actors and use cases within a system.

### View

A projection of a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective.