

Sumar

1. Setarea culorilor pentru vizualizarea acoperirii codului în IntelliJ IDEA.....	1
2. Configurarea IDE pentru rularea cu Coverage	2
3. Rularea programului și calculul acoperirii	3
4. Vizualizarea acoperirii codului sursă testat.....	4
5. Generarea și vizualizarea raportului de acoperire	5

Lista de Figuri

Figure 1. Configurarea culorilor pentru Line Coverage	2
Figure 2 Tab-ul <i>Configuration</i> în <i>Edit Configurations</i>	2
Figure 3 Tab-ul <i>Code Coverage</i> în <i>Edit Configurations</i>	3
Figure 4 Fereastra <i>Coverage</i> configurația de rulare CoverageForMedii pentru pachetul <i>repository</i>	3
Figure 5 Fereastra <i>Coverage</i> pentru clasele din pachetul <i>repository</i>	3
Figure 6. Vizualizarea acoperirii cu teste a codului rulat	4
Figure 7. Alegerea configurației de rulare care folosește JaCoCo.....	4
Figure 8. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă.....	5
Figure 9 Raportul de acoperire cu teste pentru clasele din pachetul <i>repository</i>	5
Figure 10 Raportul de acoperire cu teste pentru clasa <i>ClasaRepositoryMock</i>	5

1. Setarea culorilor pentru vizualizarea acoperirii codului în IntelliJ IDEA

- În meniul **File** ---> **Settings...** ---> **Editor** ---> **Color Scheme** ---> **General** ---> se deschide nodul **Line Coverage**;
- se setează culorile pentru cele 3 tipuri de acoperire: **Full**, **Partial**, **Uncovered**;
- se selectează **Full**, se debifează opțiunea **Foreground**; se bifează opțiunea **Background** și se setează culoarea la **CCFFCC** (verde)(vezi Figure 1), apoi **Choose**;
- similar, pentru
 - **Partial**, se bifează opțiunea **Background** și se setează culoarea la **FFFFCC** (galben) și se debifează opțiunea **Foreground**;
 - **Uncovered**, se bifează opțiunea **Background** și se setează culoarea la **FFCCCC** (roșu) și se debifează opțiunea **Foreground**.
- OK** pentru a salva setările de culoare.

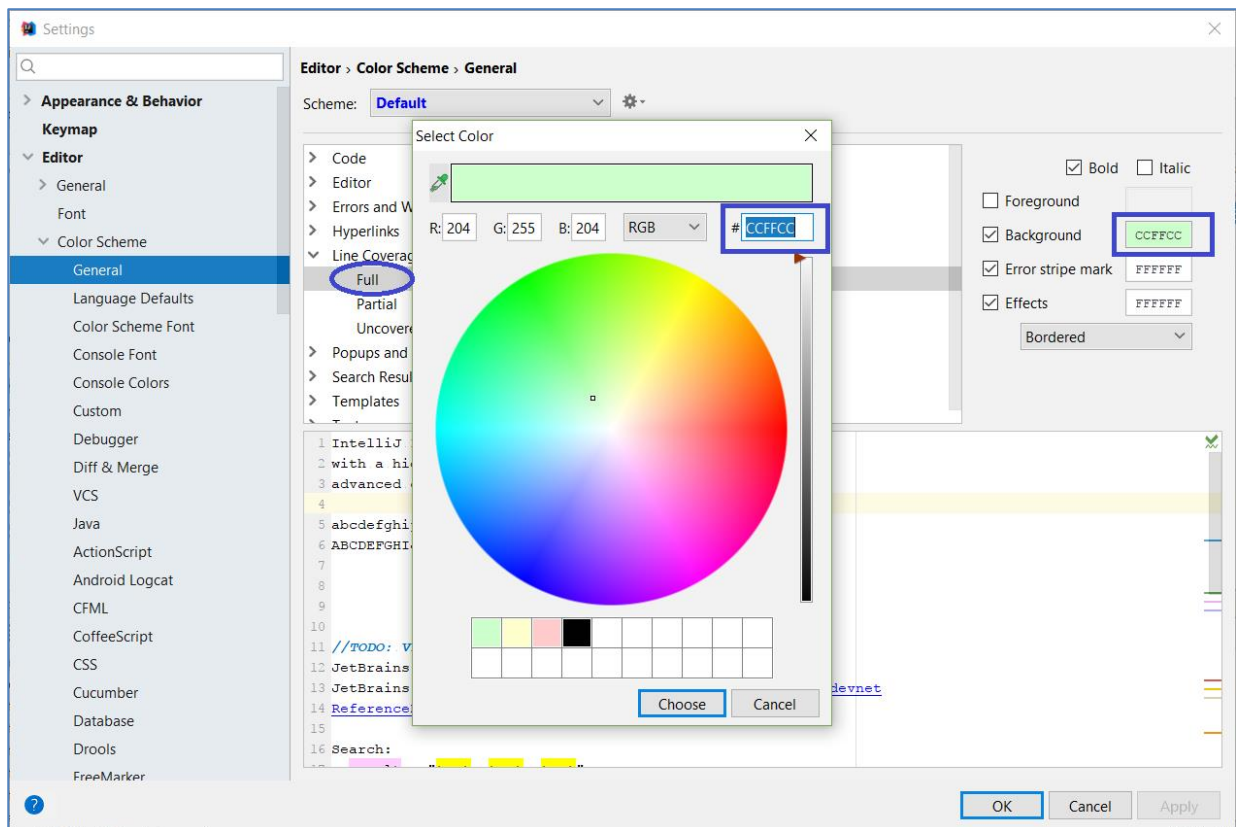


Figure 1. Configurarea culorilor pentru Line Coverage

2. Configurarea IDE pentru rularea cu Coverage

6. În meniul Run ---> **Edit Configurations...** se completează tab-urile:

- **Configuration** (vezi Figure 2):

- **Test kind:** *All in package*;
- **Package:** numele pachetului cu teste, e.g., *test*;

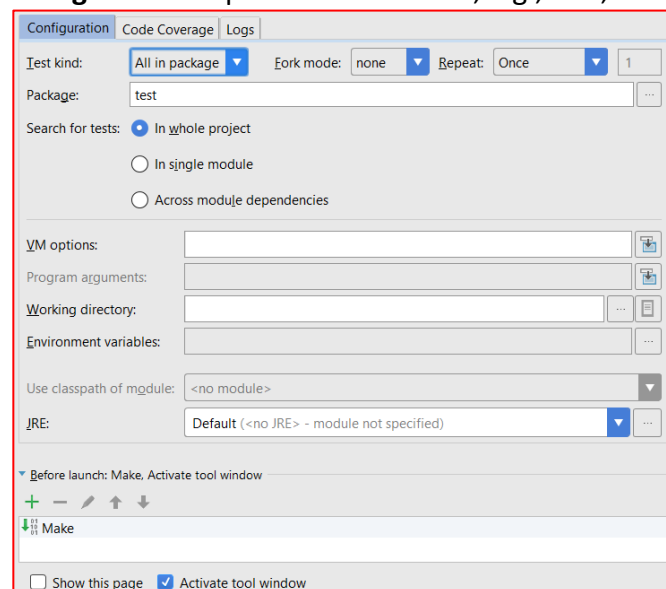

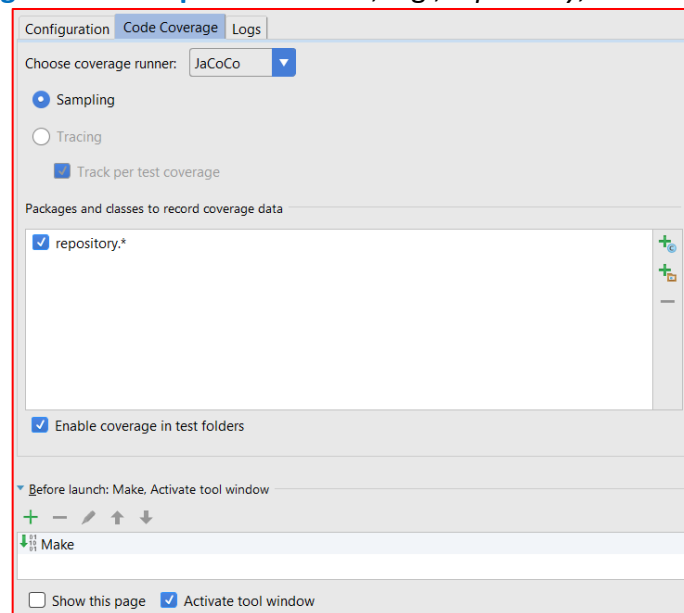


Figure 2 Tab-ul Configuration în Edit Configurations...

- **Coverage** (vezi Figure 3):

- **Choose coverage runner:** *JaCoCo* sau *IntelliJ IDEA*;

- **Packages and classes to record coverage data:** folosind butoanele  se aleg numele claselor și pachetelor pentru care se va calcula/înregistra gradul de acoperire cu teste, e.g., *repository*, *service*;

Figure 3 Tab-ul *Code Coverage* în *Edit Configurations...*

3. Rularea programului și calculul acoperirii

1. Meniul **Run** ---> **Run '[numele configurației de rulare]' with Coverage**;
2. Se va deschide fereastra **Coverage** (vezi Figure 4);
3. Pentru fiecare pachet, prin dublu-click se poate vizualiza gradul de acoperire pentru clase, metode și linii de cod (vezi Figure 5);

Element	Class, %	Method, %	Line, %
repository	100% (3/3)	73% (11/15)	69% (100/143)

Figure 4 Fereastra *Coverage* configurația de rulare *CoverageForMedii* pentru pachetul *repository*

Element	Class, %	Method, %	Line, %
ClasaRepositoryMock	100% (1/1)	66% (4/6)	86% (76/88)
EleviRepositoryMock	100% (1/1)	75% (3/4)	28% (6/21)
NoteRepositoryMock	100% (1/1)	80% (4/5)	52% (18/34)

Figure 5 Fereastra *Coverage* pentru clasele din pachetul *repository*

4. Vizualizarea acoperirii codului sursă testat

1. Meniul **Analyze** ---> **Show Coverage Data** (vezi Figure 6);
2. Se alege o configurație de rulare care bazată pe JaCoCo sau IntelliJ (vezi Figure 7);
3. În **Project Explorer**, în dreptul fiecărei entități pentru care s-a monitorizat/calculat acoperirea, se afișează procentul de acoperire (vezi Figure 8).
4. În frame-ul de editare a codului sursă, liniile de cod sursă sunt colorate corespunzător nivelului de acoperire (verde-full, galben-parțial, roșu-neacoperit) (vezi Figure 8).

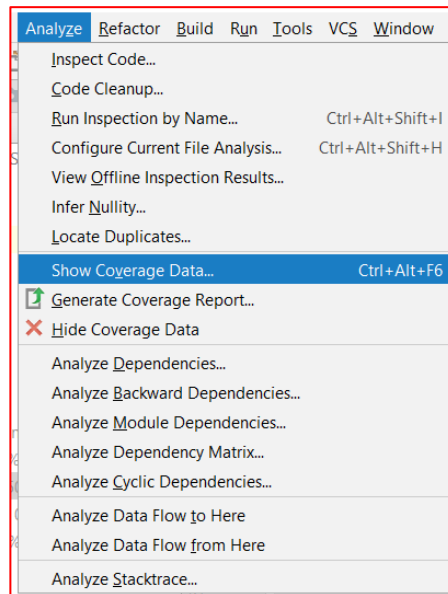


Figure 6. Vizualizarea acoperirii cu teste a codului rulat

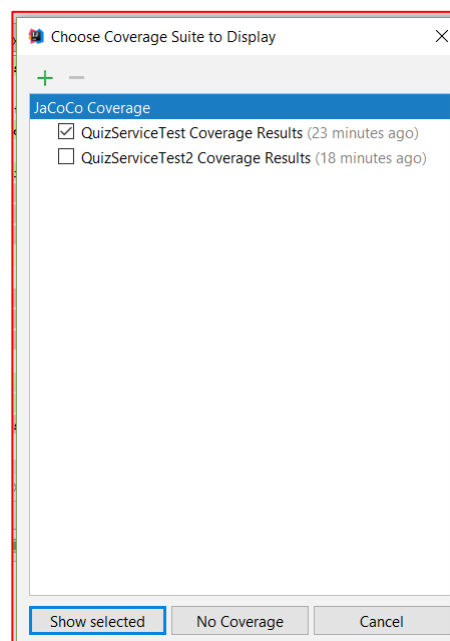


Figure 7. Alegerea configurației de rulare care folosește JaCoCo

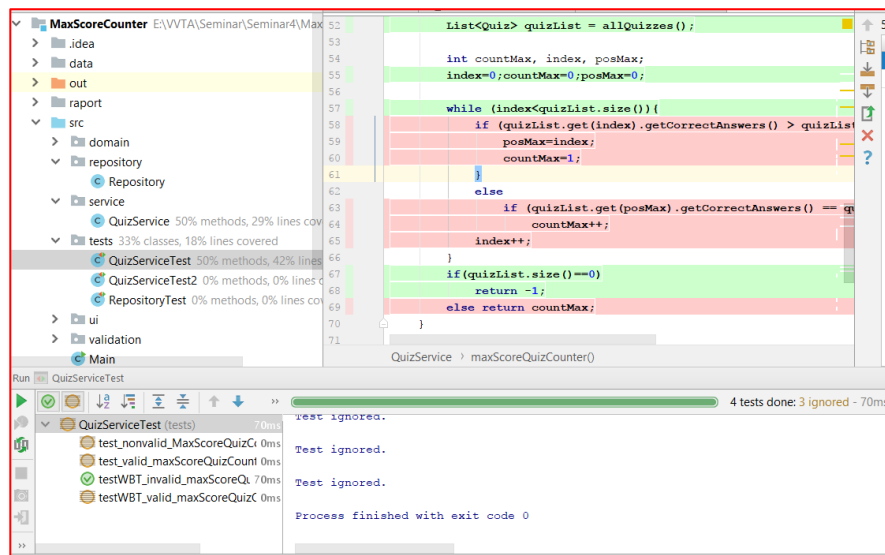


Figure 8. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă

5. Generarea și vizualizarea raportului de acoperire

1. Meniul **Analyze** ---> **Generate Coverage Report...**;
2. Se bifează opțiunea **Open generated HTML in browser** ---> **Save**;
3. Se va deschide implicit raportul de acoperire într-un browser web (vezi Figure 9);
4. Pentru fiecare clasă, prin dublu-click se poate observa **vizual** (prin culorile verde, galben, roșu) cât și **în rezumatul raportului** (prin valoarea procentului din tabel) gradul de acoperire cu teste pentru clase, metode și liniile de cod sursă (vezi Figure 10).

[all classes] [repository]

Coverage Summary for Package: repository

Package	Class, %	Method, %	Line, %
repository	100% (3/ 3)	73.3% (11/ 15)	69.9% (100/ 143)

Class	Class, %	Method, %	Line, %
ClasaRepositoryMock	100% (1/ 1)	66.7% (4/ 6)	86.4% (76/ 88)
EleviRepositoryMock	100% (1/ 1)	75% (3/ 4)	28.6% (6/ 21)
NoteRepositoryMock	100% (1/ 1)	80% (4/ 5)	52.9% (18/ 34)

Figure 9 Raportul de acoperire cu teste pentru clasele din pachetul repository

[all classes] [repository]

Coverage Summary for Class: ClasaRepositoryMock (repository)

Class	Class, %	Method, %	Line, %
ClasaRepositoryMock	100% (1/ 1)	66.7% (4/ 6)	86.4% (76/ 88)

```

1 package repository;
2
3 import java.util.ArrayList;
4 import java.util.HashMap;
5 import java.util.LinkedList;
6 import java.util.List;
7
8 import utils.ClasaException;
9 import utils.Constants;
10
11 import model.Corigent;
12 import model.Elev;
13 import model.Medie;
14 import model.Nota;
15
16 public class ClasaRepositoryMock implements ClasaRepository{
17
18     private HashMap<Elev, HashMap<String, List<Double>>> clasa;
19
20     public ClasaRepositoryMock() {
21         clasa = new HashMap<Elev, HashMap<String, List<Double>>>();
22     }
23

```

Figure 10 Raportul de acoperire cu teste pentru clasa ClasaRepositoryMock