

Sumar

1. Setarea culorilor pentru vizualizarea acoperirii codului în IntelliJ IDEA.....	1
2. Configurarea IntelliJ IDEA pentru rularea cu Coverage.....	2
3. Rularea testelor. Calculul acoperirii codului sursă testat.....	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 Coverage – configurația de rulare TaskTest pentru pachetul tasks.model.....	3
Figure 5 Fereastra <i>Coverage</i> – configurația de rulare TaskTest pentru pachetul tasks.model.....	4
Figure 6. Alegerea opțiunii pentru vizualizarea acoperirii cu teste a codului rulat	4
Figure 7. Alegerea configurației de rulare care folosește IntelliJ IDEA	5
Figure 8. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă.....	5
Figure 9. Configurarea opțiunii de generare a raportului de acoperire (deschidere în browser, folder pentru salvare)	6
Figure 10. Raportul de acoperire cu teste pentru clasele din pachetele selectate înainte de rularea testelor	6
Figure 11. Raportul de acoperire cu teste pentru clasa Task.....	6

1. Setarea culorilor pentru vizualizarea acoperirii codului în IntelliJ IDEA

- În meniul **File** ---> **Settings...** ---> **Editor** (dublu click) ---> **Color Scheme** ---> **General** ---> se deschide nodul **Line Coverage** din lista din partea dreaptă;
- aici se vor seta culorile pentru cele 3 tipuri de cod sursă explorat: **Full**, **Partial**, **Uncovered**;
- în continuare sunt descriși pașii pentru setarea culorii de **Background**, în locul celei de **Foreground**. **Totuși, fiecare echipă poate decide nivelul (Foreground sau Background) la care dorește să utilizeze codul de culori specificat (verde, galben roșu).**
- se selectează **Full**, se debifează opțiunea **Foreground**; pentru **Background** se completează culoarea **CCFFCC** (verde), apoi **Choose** (vezi Figure 1);
- similar:
 - pentru **Partial** se setează culoarea **Background** la **FFFFCC** (galben) și se debifează opțiunea **Foreground**;
 - pentru **Uncovered**, se setează culoarea **Background** la **FFCCCC** (roșu) și se debifează opțiunea **Foreground**.
- OK** pentru a salva setările referitoare la culoare.

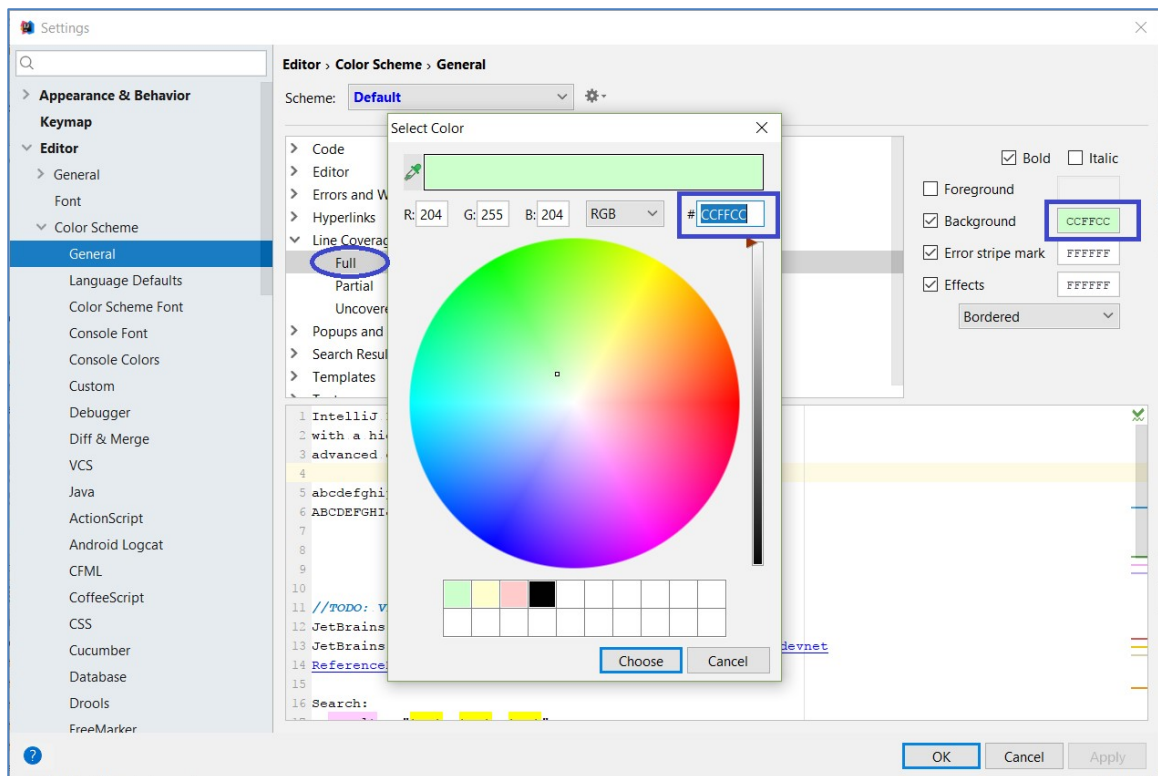


Figure 1. Configurarea culorilor pentru Line Coverage

2. Configurarea IntelliJ IDEA pentru rularea cu Coverage

1. Se alege clasa cu teste care va fi rulată;
2. În meniul **Run** ---> **Edit Configurations...** se creează o configurație de tip JUnit pentru clasa cu teste care se va rula;
3. Se completează se completează tab-urile:

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

- **Test kind:** *Class*;
- **Class:** numele clasei cu teste, e.g., *tasks.TaskTest*;

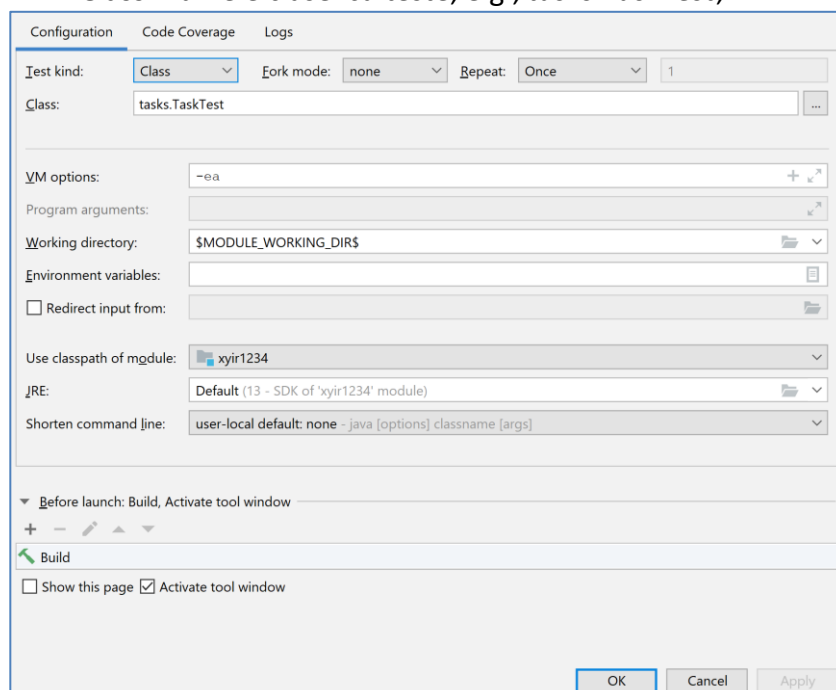



Figure 2 Tab-ul Configuration în Edit Configurations...

- **Code Coverage** (vezi Figure 3):
 - Choose coverage runner: *IntelliJ IDEA*;
 - Packages and classes to record coverage data: folosind butoanele  se aleg numele claselor și/sau pachetelor pentru care se va calcula/înregistra gradul de acoperire la rularea testelor, e.g., *tasks*;

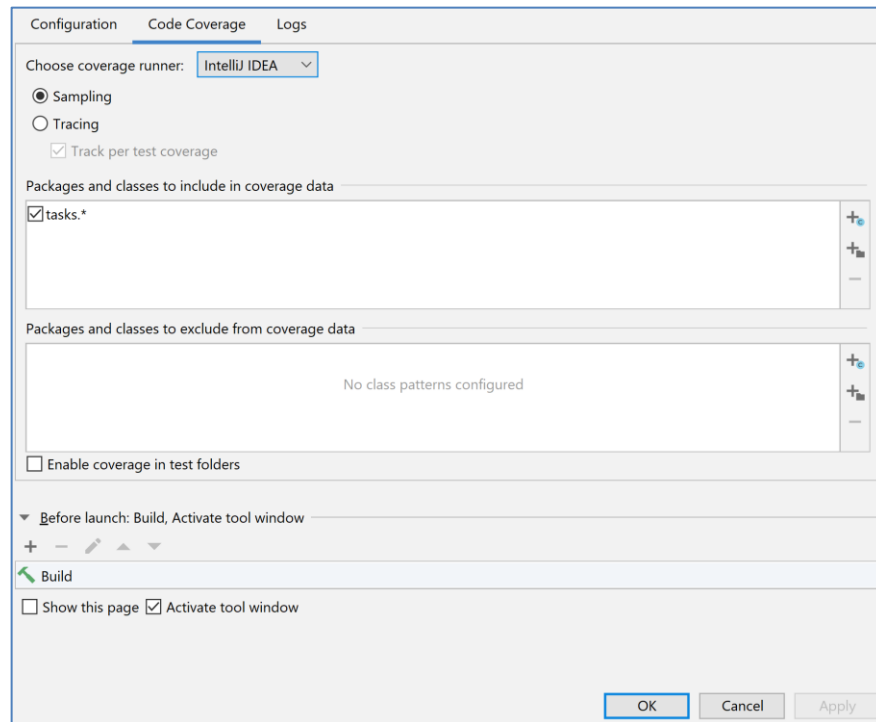


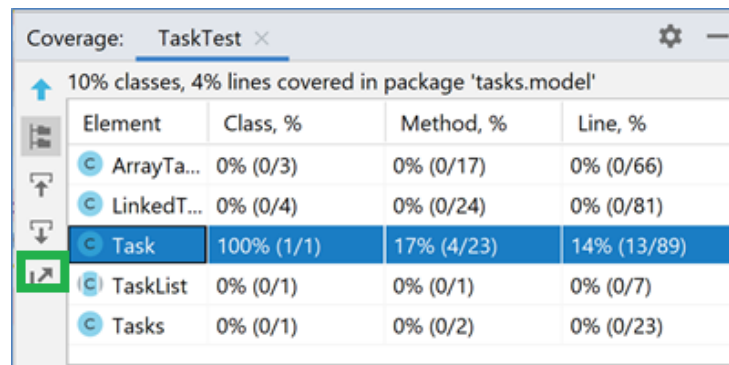
Figure 3 Tab-ul Code Coverage în Edit Configurations...

3. Rularea testelor. Calculul acoperirii codului sursă testat

1. Meniul **Run** ---> **Run '[numele configurației de rulare]' with Coverage** sau în **Project Explorer**, click dreapta pe numele clasei cu teste și se alege opțiunea **Run '[numele configurației de rulare]' with Coverage**;
2. După rulare 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);

Coverage: TaskTest × TaskTest ×			
7% classes, 2% lines covered in 'all classes in scope'			
Element	Class, %	Method, %	Line, %
tasks.model	10% (1/10)	7% (5/70)	4% (12/279)
tasks.services	0% (0/3)	0% (0/27)	0% (0/220)

Figure 4. Fereastra Coverage – configurația de rulare TaskTest pentru pachetul tasks.model



Element	Class, %	Method, %	Line, %
ArrayTa...	0% (0/3)	0% (0/17)	0% (0/66)
LinkedT...	0% (0/4)	0% (0/24)	0% (0/81)
Task	100% (1/1)	17% (4/23)	14% (13/89)
TaskList	0% (0/1)	0% (0/1)	0% (0/7)
Tasks	0% (0/1)	0% (0/2)	0% (0/23)

Figure 5 Fereastra Coverage – configurația de rulare TaskTest pentru pachetul tasks.model

4. Vizualizarea acoperirii codului sursă testat

1. Meniul **Run** ---> **Show Code Coverage Data** (vezi Figure 6);
2. Se alege o configurație de rulare care e bazată pe *JaCoCo* sau *IntelliJ IDEA* (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. după execuția testelor, 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).

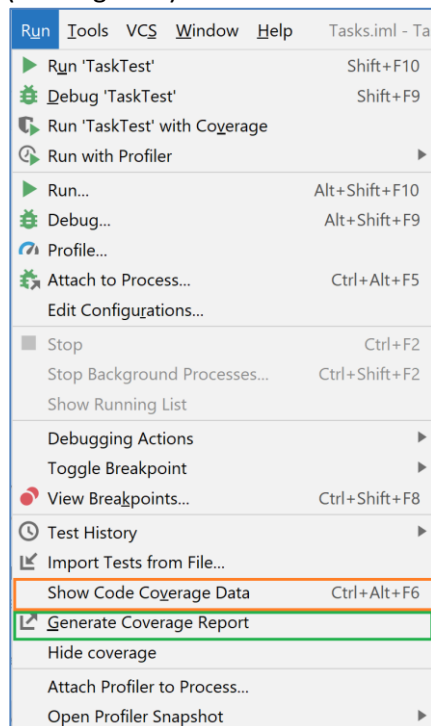


Figure 6. Alegerea opțiunii pentru vizualizarea acoperirii cu teste a codului rulat

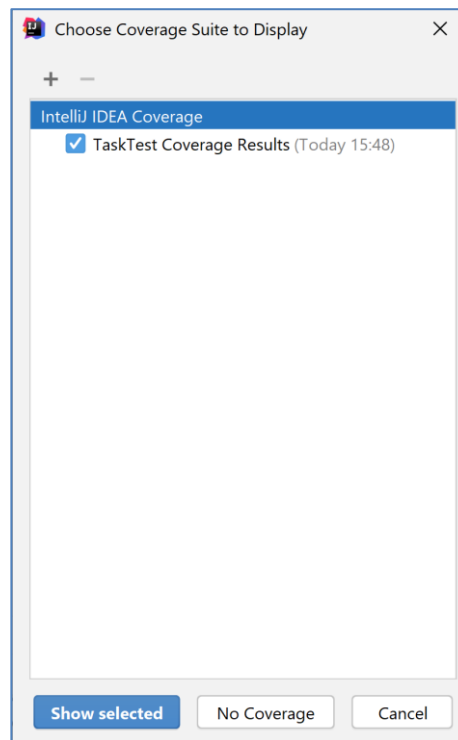


Figure 7. Alegerea configurației de rulare care folosește IntelliJ IDEA

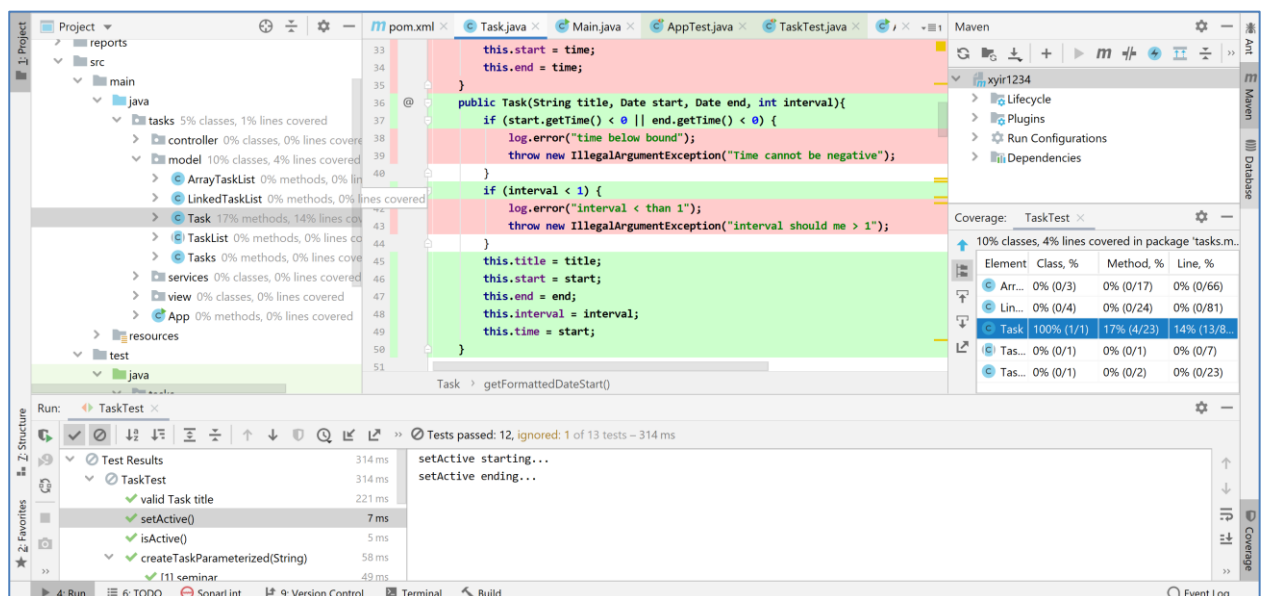


Figure 8. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă

5. Generarea și vizualizarea raportului de acoperire

Accesarea opțiunii de generare a raportului de acoperire se poate face în două moduri:

Varianta 1:

- Meniul Run ---> **Generate Coverage Report** (vezi Figure 6);

Varianta 2:

- Fereastra **Coverage**, butonul marcat cu verde (**Generate Coverage Report**) (vezi Figure 5);

Apoi:

1. Se bifează opțiunea **Open generate HTML in browser**;
2. Se alege folderul în care se salvează raportul, apoi **Save** (vezi Figure 9);
3. Se va deschide implicit raportul de acoperire într-un browser web (vezi Figure 10);
4. Pentru fiecare clasă, prin dublu-click se poate observa (**visual**, 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 11).

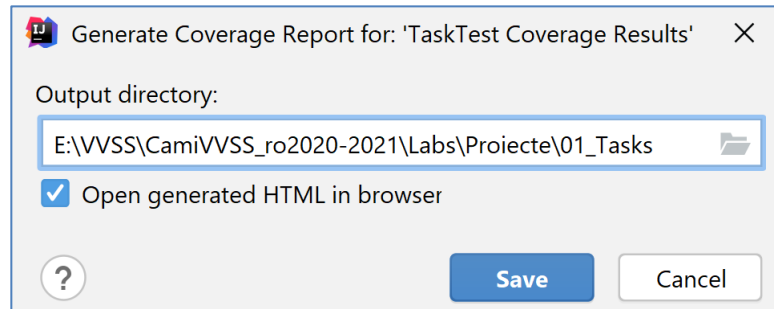


Figure 9. Configurarea opțiunii de generare a raportului de acoperire (deschidere in browser, folder pentru salvare)

[all classes]

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	9.1% (1/ 11)	5.1% (5/ 99)	2.5% (12/ 480)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
tasks.model	12.5% (1/ 8)	7% (5/ 71)	4.4% (12/ 270)
tasks.services	0% (0/ 3)	0% (0/ 28)	0% (0/ 210)

generated on 2021-03-19 18:30

Figure 10. Raportul de acoperire cu teste pentru clasele din pachetele selectate înainte de rularea testelor

[all classes] [tasks.model]

Coverage Summary for Class: Task (tasks.model)

Class	Class, %	Method, %	Line, %
Task	100% (1/ 1)	21.7% (5/ 23)	13.5% (12/ 89)

```

1 package tasks.model;
2
3 import org.apache.log4j.Logger;
4 import tasks.services.TaskIO;
5
6 import java.io.Serializable;
7 import java.text.SimpleDateFormat;
8 import java.util.Date;
9
10
11 public class Task implements Serializable, Cloneable {
12     private String title;
13     private Date time;
14     private Date start;
15     private Date end;
16     private int interval;
17     private boolean active;
18
19     private static final Logger log = Logger.getLogger(Task.class.getName());
20     private static final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");
21
22     public static SimpleDateFormat getDateFormat(){
23         return sdf;
24     }
25     public Task(String title, Date time){
26         if (time.getTime() < 0) {
27             log.error("time below bound");
28             throw new IllegalArgumentException("Time cannot be negative");
29         }
30         this.title = title;
31         this.time = time;
32         this.start = time;
33         this.end = time;

```

Figure 11. Raportul de acoperire cu teste pentru clasa Task