

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.....	4
4. Vizualizarea acoperirii codului sursă testat.....	4
5. Generarea și vizualizarea raportului de acoperire	6

Lista de Figuri

Figure 1. Configurarea culorilor pentru Line Coverage	2
Figure 2 Alegerea clasei cu teste	2
Figure 3 Configurarea opțiunilor de acoperire	3
Figure 4. Configurarea opțiunilor de acoperire (forma finală)	3
Figure 5. Fereastra Coverage – configurația de rulare <i>TaskTest</i> pentru clasa <i>Task</i> din pachetul <i>model</i>	4
Figure 6 Fereastra <i>Coverage</i> – configurația de rulare <i>TaskTest</i> pentru pachetul <i>tasks.model</i>	4
Figure 7. Alegerea opțiunii pentru vizualizarea acoperirii cu teste a codului rulat	5
Figure 8. Alegerea configurației de rulare care folosește IntelliJ IDEA	5
Figure 9. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă.....	6
Figure 10. Configurarea opțiunii de generare a raportului de acoperire (deschidere în browser, folder pentru salvare)	6
Figure 11. Raportul de acoperire cu teste pentru clasele din pachetele selectate după de rularea testelor	7
Figure 12. Raportul de acoperire cu teste pentru clasa <i>Task</i>	7

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 explorare a codului sursă: **Full**, **Partial**, **Uncovered**;
- în continuare sunt descriși pașii pentru setarea culorii de **Background**, în locul celei de **Foreground**. **Fiecare echipă poate decide nivelul (Foreground sau Background) pentru 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.

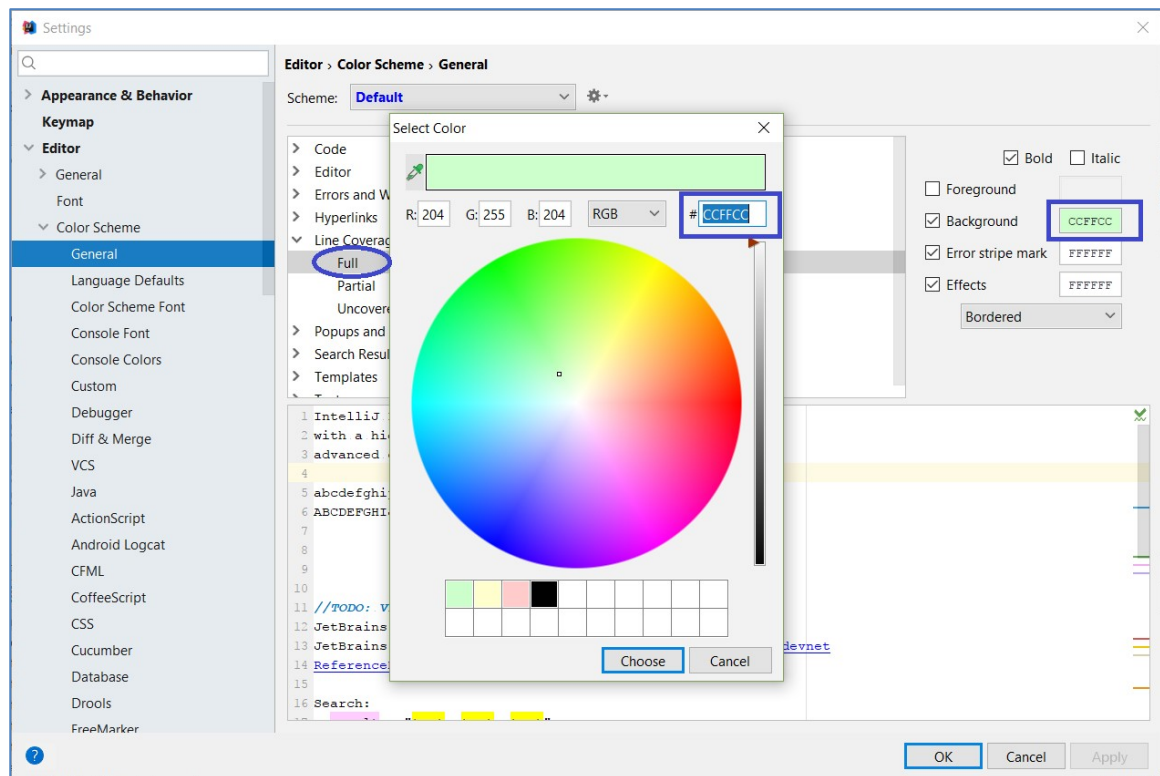


Figure 1. Configurarea culorilor pentru Line Coverage

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 rulare de tip JUnit;
3. Se completează:
 - **Alegerea clasei de test** (vezi Figure 2):
 - **Test kind**: *Class*;
 - **Class**: numele clasei cu teste, e.g., *tasks.TaskTest*;
 - se alege opțiunea **Modify Options** pentru configurarea acoperii la execuția testelor;

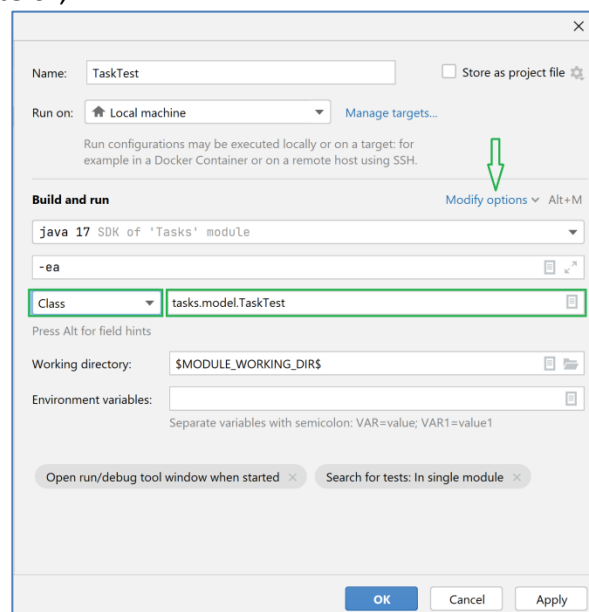


Figure 2 Alegerea clasei cu teste

• **configurarea opțiunilor de acoperire** (vezi Figure 3, Figure 4):

- **Specify classes and packages** – se aleg clasele/pachetele pentru care să se realizeze (înregistreze sau calculeze) acoperirea la testare, e.g., clasa **Task**, pachetul **repository**;
- **Specify alternative coverage runner** – se alege opțiunea **IntelliJ IDEA**;

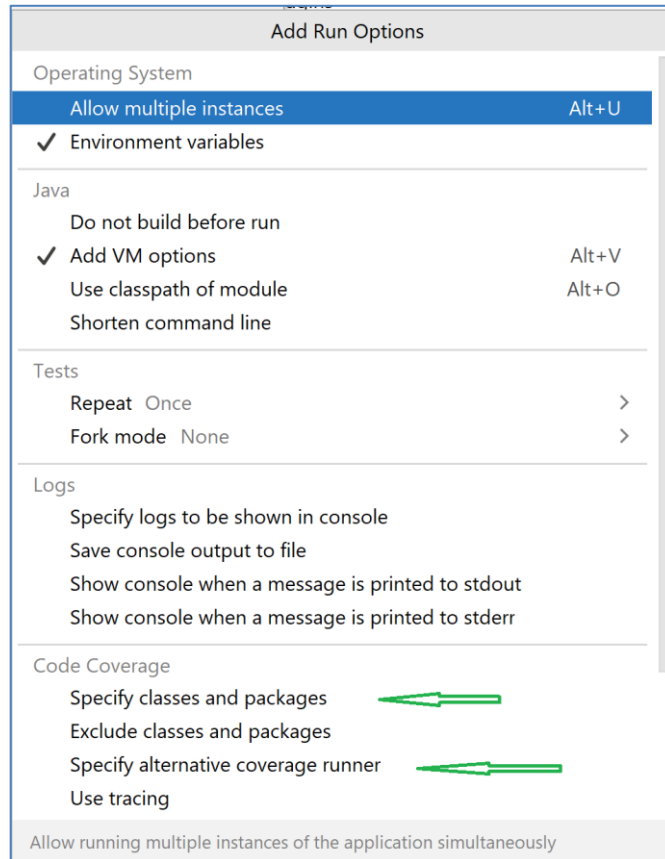


Figure 3 Configurarea opțiunilor de acoperire

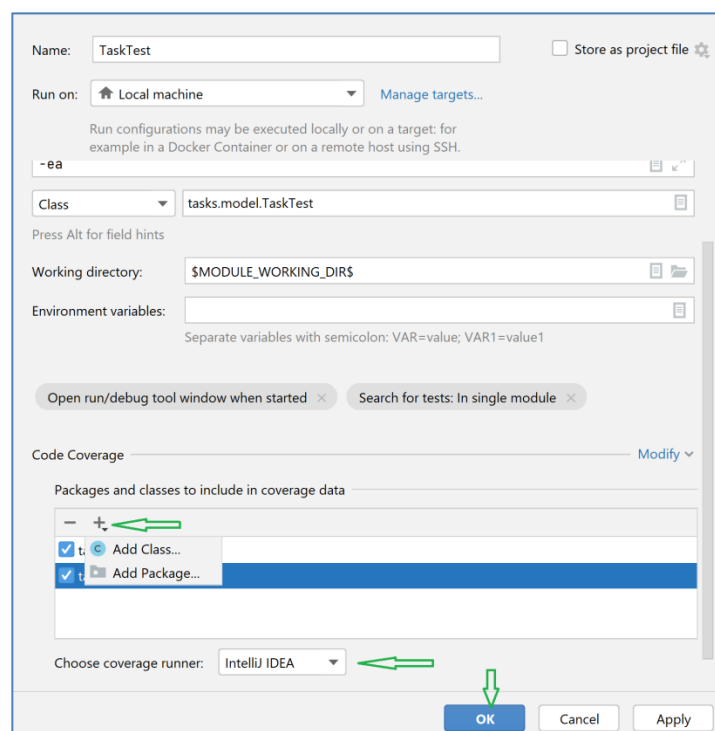


Figure 4. Configurarea opțiunilor de acoperire (forma finală)

3. Rularea testelor. Calculul acoperirii codului sursă testat

1. Meniul **Run** ---> **Run '[numele configurației de rulare]' with Coverage**;
2. După rulare se va deschide fereastra **Coverage** (vezi Figure 5);
3. Pentru fiecare pachet, prin dublu-click se poate vizualiza gradul de acoperire pentru clase, metode și linii de cod (vezi Figure 5, Figure 6);

Element	Class, %	Method, %	Line, %
tasks.services	0% (0/3)	0% (0/27)	0% (0/220)
tasks.model.Task	100% (1/1)	21% (5/23)	13% (11/83)

Figure 5. Fereastra Coverage – configurația de rulare TaskTest pentru clasa Task din pachetul model

Element	Class, %	Method, %	Line, %
DataService	0% (0/1)	0% (0/5)	0% (0/20)
TaskIO	0% (0/1)	0% (0/16)	0% (0/178)
TasksService	0% (0/1)	0% (0/6)	0% (0/22)

Figure 6 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 7);
2. Se alege o configurație de rulare care e bazată pe *IntelliJ IDEA* (vezi Figure 8), apoi se alege **Show selected**;
3. În **Project Explorer**, în dreptul fiecărei entități pentru care s-a monitorizat/calculat acoperirea, se afișează procentul de acoperire (vezi Figure 9);
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 9).

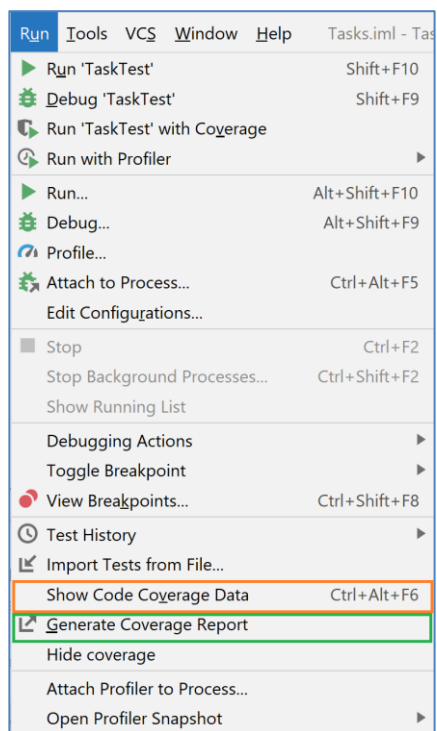


Figure 7. Alegerea opțiunii pentru vizualizarea acoperirii cu teste a codului rulat

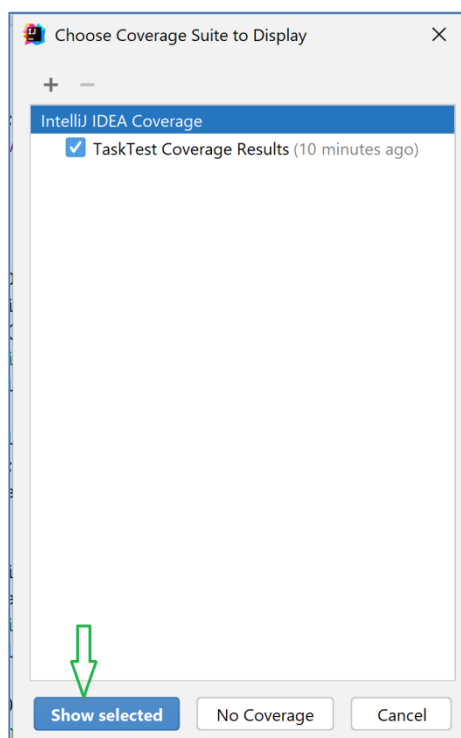


Figure 8. Alegerea configurației de rulare care folosește IntelliJ IDEA

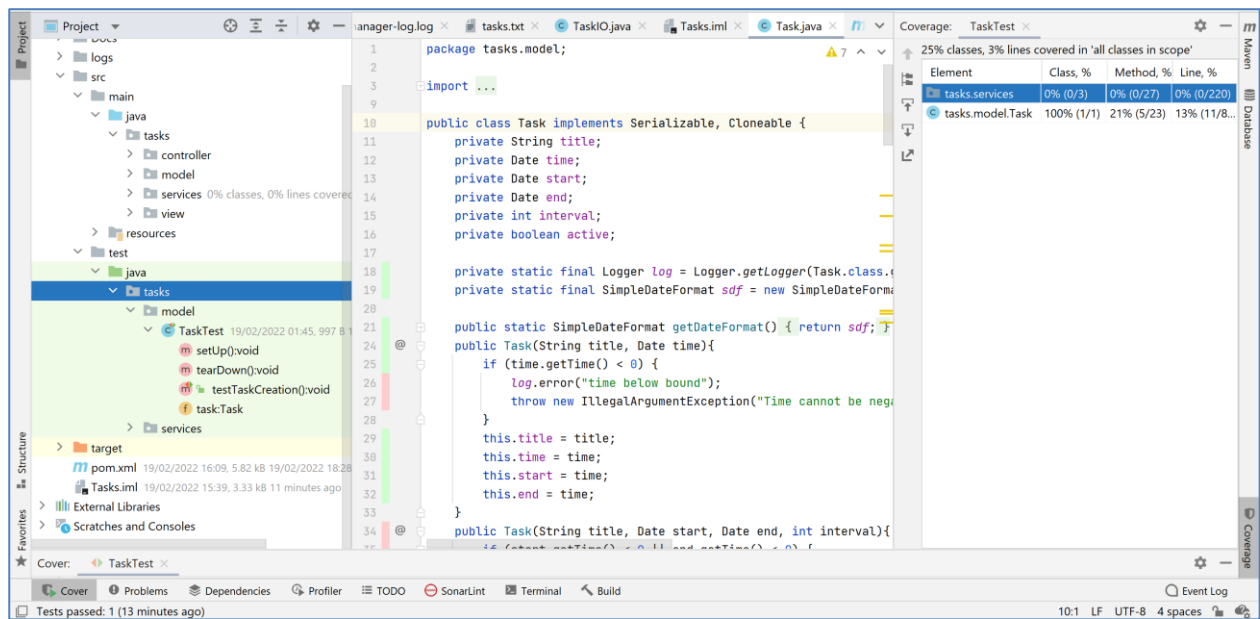


Figure 9. Vizualizarea acoperirii în procente și culori pentru configurația de rulare aleasă

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 7);

Varianta 2:

- Fereastra **Coverage**, butonul marcat cu verde (**Generate Coverage Report**) (vezi Figure 6).

Apoi:

1. Se bifează opțiunea **Open generate HTML in browser**;
2. Se alege folderul în care se salvează raportul, apoi **Save** (vezi Figure 10);
3. Se va deschide implicit raportul de acoperire într-un browser web (vezi Figure 11);
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 12).

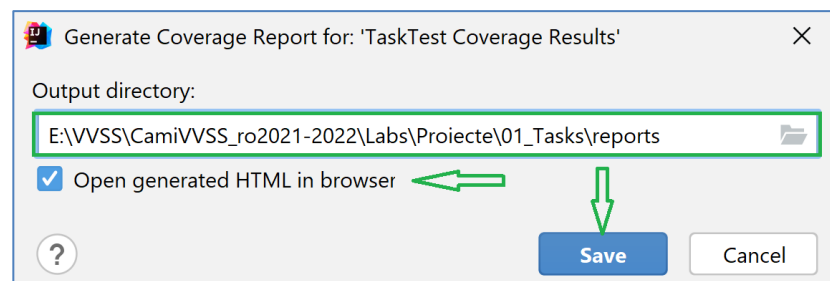


Figure 10. Configurarea opțiunii de generare a raportului de acoperire (deschidere in browser, folder pentru salvare)

Current scope: all classes

Overall Coverage Summary			
Package	Class, %	Method, %	Line, %
all classes	25% (1/4)	9.8% (5/51)	3.8% (11/293)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
tasks.model	100% (1/1)	21.7% (5/23)	13.3% (11/83)
tasks.services	0% (0/3)	0% (0/28)	0% (0/210)

generated on 2022-03-13 00:11

Figure 11. Raportul de acoperire cu teste pentru clasele din pachetele selectate după de rularea testelor

Current scope: all classes | tasks.model

Coverage Summary for Class: Task (tasks.model)			
Class	Class, %	Method, %	Line, %
Task	100% (1/1)	21.7% (5/23)	13.3% (11/83)

```

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 public class Task implements Serializable, Cloneable {
11     private String title;
12     private Date time;
13     private Date start;
14     private Date end;
15     private int interval;
16     private boolean active;
17
18     private static final Logger log = Logger.getLogger(Task.class.getName());
19     private static final SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");
20
21     public static SimpleDateFormat getDateFormat() {
22         return sdf;
23     }
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;
34     }
35
36     public Task(String title, Date start, Date end, int interval) {
37         if (start.getTime() < 0 || end.getTime() < 0) {
38             log.error("Time below bound");
39             throw new IllegalArgumentException("Time cannot be negative");
40         }
41         if (interval < 1) {
42             log.error("Interval < than 1");
43             throw new IllegalArgumentException("Interval should be > 1");
44         }
45         this.title = title;

```

Figure 12. Raportul de acoperire cu teste pentru clasa Task