**Cerințe obligatorii**

1. Pattern-urile implementate trebuie sa respecte definitia din GoF discutată în cadrul cursurilor și laboratoarelor. Nu sunt acceptate variații sau implementării incomplete.
2. Pattern-ul trebuie implementat corect în totalitate corect pentru a fi luat în calcul
3. Soluția nu conține erori de compilare
4. Testele unitare sunt considerate corecte doar dacă sunt implementate conform cerințelor și dacă metodele sunt corectate corespunzător pe baza lor
5. Pattern-urile pot fi tratate distinct sau pot fi implementate pe același set de clase

**Cerințe Clean Code obligatorii (soluția este depunctata cu câte 5 puncte pentru fiecare cerința ce nu este respectată**) - maxim se pot pierde 15 puncte

1. Pentru denumirea claselor, funcțiilor, testelor unitare, atributelor și a variabilelor se respecta convenția de nume de tip Java Mix CamelCase;
2. Pattern-urile, Test Case-urile, Excepțiile și clasa ce contine metoda main() sunt definite in pachete distincte ce au forma *cts.nume.prenume.gNrGrupa.teste*, *cts.nume.prenume.gNrGrupa.patternX*, *cts.nume.prenume.gNrGrupa.main* (studenții din anul suplimentar trec “as” în loc de gNrGrupa)
3. Clasele și metodele sunt implementate respectând principiile KISS, DRY și SOLID (atenție la DIP)

Se dezvoltă o aplicație software pentru editarea imaginiilor.

**10p.** Dezvolți o soluție software pentru editarea imaginilor de tip Bitmap destinată copiilor. Soluția trebuie sa le permită utilizatorilor sa vadă istoricul modificărilor si să le permită revenirea la un moment in timp ales de ei. Imaginile ce pot fi procesate în cadrul aplicației sunt de tip ***ImagineBitmap* (clasa trebuie completata cu încă 1 atribut la alegere si încă 2 metode)**. Soluția software trebuie să permită stocarea salvărilor și trebuie sa permită utilizatorului să le vadă și să le recupereze în funcție de data salvării (aceasta informație este generata automat în momentul în care se face salvarea imaginii si este adăugată in meta-datele salvării)

**5p.** Pattern-ul este testat în main() prin crearea unei imagini care este apoi modificată de cel puțin 3 ori prin metodele din clasă. După fiecare modificare imaginea este afișată și salvată iar aceasta salvare este gestionata de către istoricul modificărilor. La final să se restaureze imaginea la momentul inițial si sa se afișeze.

**5p.** În timp soluția aceasta este achiziționată de o altă companie care are un produs similar dar care permite procesarea de imagini de tip ***PhotoImage*.** Ca parte a vânzării trebuie găsită o soluție care să permite utilizarea imaginilor Bitmap în noul context.

**5p.** Să se testeze soluția prin definirea unui obiect de tip ***ImagineBitmap*** și să se demonstreze că acesta poate fi integrat și folosit în noul context prin soluția oferită.

**6p.** Dându-se clasa *TestGrila* și restricțiile definite prin comentarii să se implementeze teste unitare (toate testele unitare aferente unei metode sunt implementate într-un TestCase) care să cuprindă:

1. un unit test care să realizeze o testare *Right* pentru **setDificultate()** **(1.5p)**
2. un unit test care să testeze o testare *Error Condition* pentru **setPunctajMaxim() (1.5p)**
3. un unit test de tip *Ordering* pentru metoda ***calculTotalPuncte()* (1.5p)**
4. un unit test de verificare de tip *Cardinality = 1* pentru metoda ***calculTotalPuncte*()**;**(1.5p)**

**2p.** Să se implementeze o suită de teste care să conțină DOAR câte o metodă, la alegere, din fiecare test case

**2p.** Prin testele implementate sau prin adăugarea de teste noi sa se testeze *setDificultate* ()asigurând un code coverage de 100% pentru această metodă.