5/8/2019

ISP – tema 2

Generarea de coduri castigatoare ale unui concurs

Mihai Badea, Ana Grosu, Tudor Mangri

332AB, ingineria sistemelor, automatica si calculatoare

Cerinta

Sistem pentru generarea random a premiilor la un concurs de ciocolata. Sunt trei categorii de premii, cu un nr fix de premii in fiecare categorie. La inceputul concursului se genereaza random o data si o ora pentru fiecare premiu din fiecare categorie, intr-un anumit interval. Fiecare utilizator incarca unul sau mai multe coduri. La fiecare cod incarcat el va participa la o prima categorie, la fiecare 3 coduri la a doua categorie, si la fiecare 5 coduri la cea de-a treia. Imediat dupa introducerea unui nou cod va primi mesaj daca acel cod a fost castigator sau nu.

Lucru in echipa:

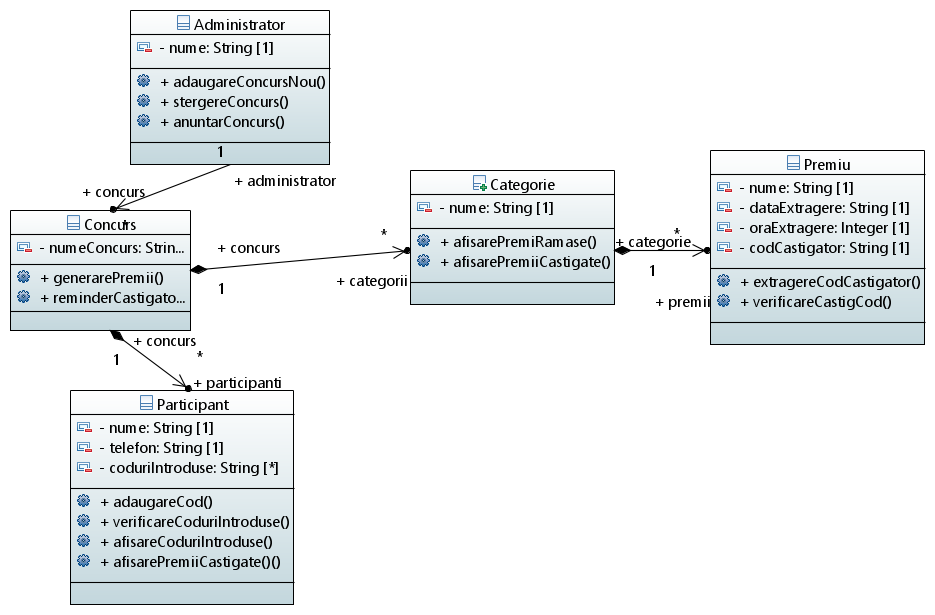
1. Implementati clasele conform detaliilor precizate in diagrama de clase UML (de la Tema 1, actualizata daca este nevoie)
2. Implementati de asemenea relatiile dintre clase, conform diagramei de clase UML. La predare trebuie de asemenea prezentata aceasta diagrama.

Lucru individual:

1. Fiecare student trebuie sa implementeze un test folosind Junit, care sa verifice ca s-a executat cu succes cazul de utilizare descris la Tema 1 prin intermediul diagramei de activitati. Testul trebuie sa prezinte o functionalitate specifica temei si sa se execute automat, pentru mai multe seturi de date de test (intrari + iesiri asteptate) – in asa fel incat sa fie verificate toate situatiile posibile (parcurgerea tuturor buclelor, fortarea unor exceptii, limitele inferioare/superioare ale tablourilor etc)

Lucru in echipa

Cerinta 1

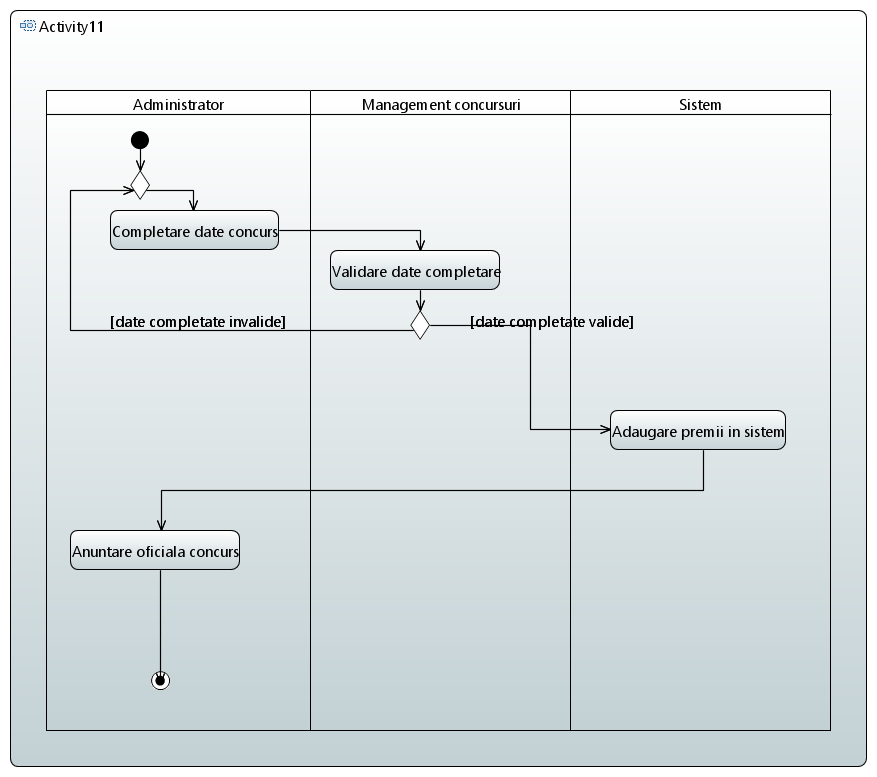


Lucru individual

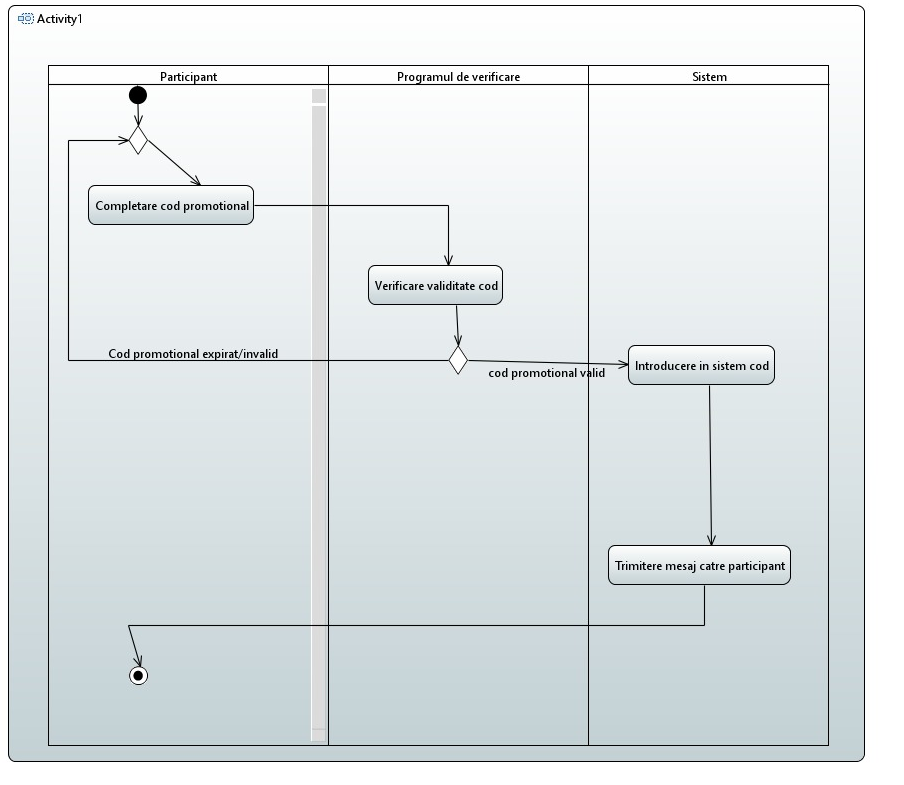
Cerinta 3

Mai jos se gasesc diagramele pe baza carora am implementat testele Junit.

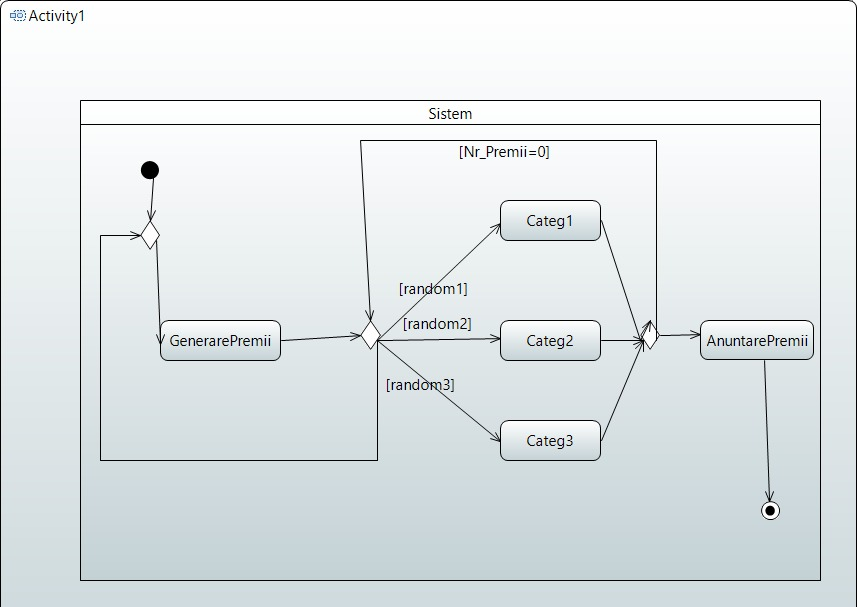
*Adaugarea unui nou concurs – Ana Grosu*



*Adaugarea unui cod promotional – Mihai Badea*



*Generarea de premii – Tudor Mangri*



Anexa

*Functiile de verificare– Ana Grosu*

|  |  |
| --- | --- |
| public boolean verificareNumeConcurs() {  if (this.numeConcurs.equals("")) {  return false;  };  if (this.numeConcurs.length() > 30) {  return false;  };  return true;  }    public boolean verificareExistentaPremii() {  if (this.categorii[0].premii.length == 0) {  return false;  };  if (this.categorii[1].premii.length == 0) {  return false;  };  if (this.categorii[2].premii.length == 0) {  return false;  };  return true;  }    public boolean verificareExistentaDataExtragere() {  for(int i = 0; i < 3; i ++) {  for (int j = 0; j < this.categorii[i].premii.length; j++) {  if (this.categorii[i].premii[j].getDataExtragere().equals("")) {  return false;  };  }  }  return true;  } | public boolean validareOraData() {  for(int i = 0; i < 3; i ++) {  for (int j = 0; j < this.categorii[i].premii.length; j++) {  int ora = is.categorii[i].premii[j].getOraExtragere();  String data = this.categorii[i].premii[j].getDataExtragere();  if(ora<0 || ora > 24) return false;    int zi = Integer.parseInt( data.split("\\.")[0]);  int luna = Integer.parseInt( data.split("\\.")[1]);    if(zi<1 || zi > 31) return false;  if(luna<1 || luna > 12) return false;    };  }  return true;  } |

*Testele JUnit– Ana Grosu*

|  |  |
| --- | --- |
| package TicheteConcursCiocolata;  import static org.junit.jupiter.api.Assertions.\*;  import org.junit.jupiter.api.Test;    class testareAdaugareConcurs {        @Test      void testVerificareNumeConcurs() {          //initializari          Premiu[] premii = new Premiu[1];          premii[0] = new Premiu();          Categorie[] categorii = new Categorie[1];          categorii[0] = new Categorie("", premii);            //testam numele de concurs gol          Concurs concurs1 = new Concurs("", categorii);          assertFalse(concurs1.verificareNumeConcurs());            //testam un nume de concurs corect          Concurs concurs2 = new Concurs("numeleMeu", categorii);          assertTrue(concurs2.verificareNumeConcurs());            //testam cu un nume prea lung          Concurs concurs3 = new Concurs("numeleMeunumeleMeunumeleMeunumeleMeu", categorii);          assertFalse(concurs3.verificareNumeConcurs());      }        @Test      void testVerificareExistentaPremii() {          //initializari          Premiu[] premii = new Premiu[1];          Premiu[] premiiGoale = new Premiu[0];          premii[0] = new Premiu();            Categorie[] categorii = new Categorie[3];          categorii[0] = new Categorie("", premii);          categorii[1] = new Categorie("", premiiGoale);          categorii[2] = new Categorie("", premii);            //testam cu categorii ce nu contin premii          Concurs concurs1 = new Concurs("", categorii);          assertFalse(concurs1.verificareExistentaPremii());            //testam cu categorii ce contin premii          categorii[1] = new Categorie("", premii);          Concurs concurs2 = new Concurs("", categorii);          assertTrue(concurs2.verificareExistentaPremii());      } | @Test      void testVerificareExistentaDataExtragere() {          //initializari          Premiu[] premiiFaraData = new Premiu[1];          premiiFaraData[0] = new Premiu();            Premiu[] premiiCuData = new Premiu[1];          premiiCuData[0] = new Premiu("premiu","12.12",0,"cod");            Categorie[] categoriiFaraData = new Categorie[3];          categoriiFaraData[0] = categoriiFaraData[1] =categoriiFaraData[2]= new Categorie("", premiiFaraData);            Categorie[] categoriiCuData = new Categorie[3];          categoriiCuData[0] = categoriiCuData[1] =categoriiCuData[2] = new Categorie("", premiiCuData);            //testam cu un concurs in care un premiu nu are data de extragere          Concurs concurs1 = new Concurs("", categoriiFaraData);          assertFalse(concurs1.verificareExistentaDataExtragere());            //testam cu un concurs in care exista data de extragere          Concurs concurs2 = new Concurs("", categoriiCuData);          assertTrue(concurs2.verificareExistentaDataExtragere());        } |

|  |  |
| --- | --- |
| @Test      void testValidareOraData() {            Premiu[] premii1 = new Premiu[1];          Premiu[] premii2 = new Premiu[1];          Premiu[] premii3 = new Premiu[1];          Premiu[] premii4 = new Premiu[1];            premii1[0] = new Premiu("premiu","12.12",234,"cod");          premii2[0] = new Premiu("premiu","1232.12",13,"cod");          premii3[0] = new Premiu("premiu","12.1232",13,"cod");          premii4[0] = new Premiu("premiu","12.12",13,"cod");            Categorie[] categorii1 = new Categorie[3];          Categorie[] categorii2 = new Categorie[3];          Categorie[] categorii3 = new Categorie[3];          Categorie[] categorii4 = new Categorie[3];            categorii1[0] = categorii1[1] =categorii1[2]= new Categorie("", premii1);          categorii2[0] = categorii2[1] =categorii2[2]= new Categorie("", premii2);          categorii3[0] = categorii3[1] =categorii3[2]= new Categorie("", premii3);          categorii4[0] = categorii4[1] =categorii4[2]= new Categorie("", premii4);            //testam cu un ora invalida          Concurs concurs1 = new Concurs("", categorii1);          assertFalse(concurs1.validareOraData());            //testam cu un zi invalida          Concurs concurs2 = new Concurs("", categorii2);          assertFalse(concurs2.validareOraData());            //testam cu un luna invalida          Concurs concurs3 = new Concurs("", categorii3);          assertFalse(concurs3.validareOraData());            //testam cu un data corecta          Concurs concurs4 = new Concurs("", categorii4);          assertTrue(concurs4.validareOraData());      }    } |  |