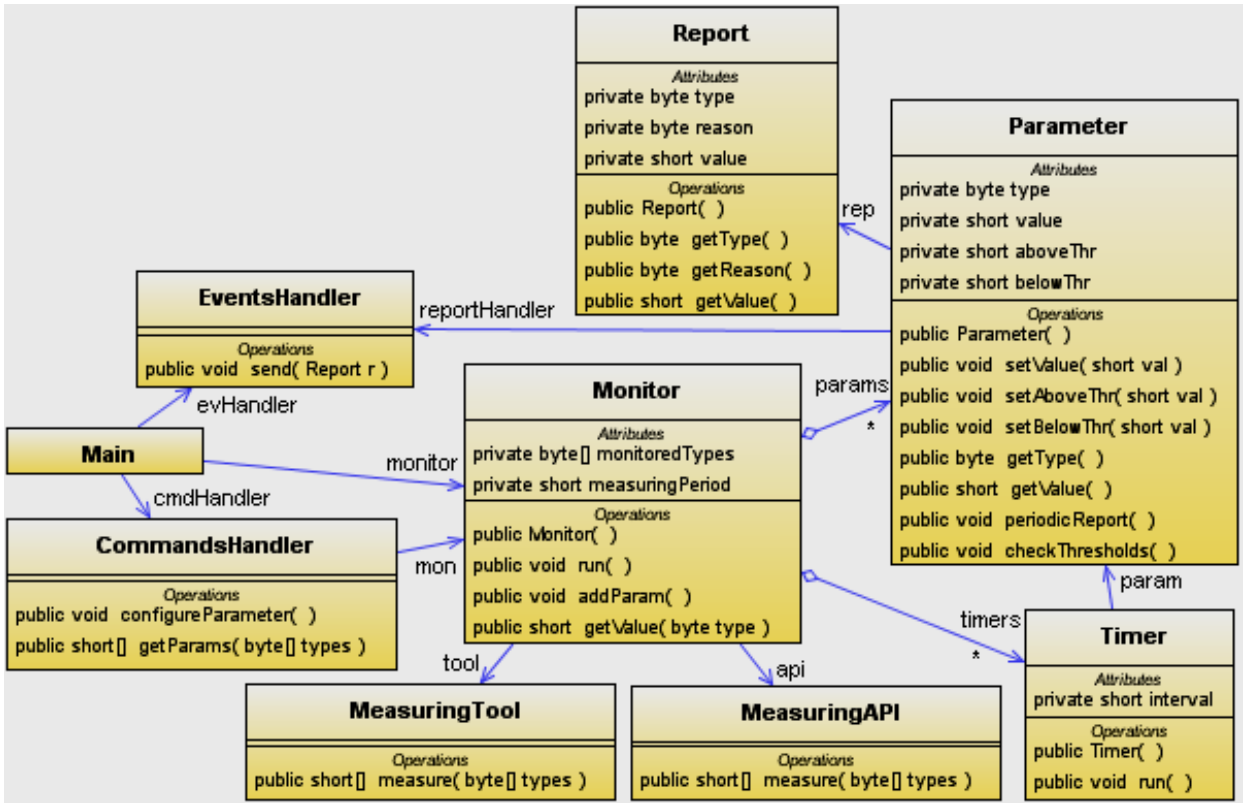
09.02.2009 **Subiect tip A** Se da **diagrama UML de clase**:



Clasa **Main** creeaza obiectele din clasa **Monitor** (care extinde **Thread**), **EventsHandler** si **CommandsHandler** si face legaturile dintre ele.

Constructorul **Monitor()** primeste **perioada efectuarii masuratorilor**, un **flag care indica metoda de masurare** (**Tool** sau **API**) si o **referinta catre un obiect** de tip **EventsHandler** (care va fi transmisa obiectelor de tip **Parameter** si **Report** in momentul crearii acestora), **sa creeze spatiul necesar tratarii a 50 de parametri**, si **sa lanseze firul de executie** al obiectului **Monitor**.

1. Sa se scrie codul constructorului **Monitor()**. **(10 pct.)**

Apelul **configureParameter()** primeste **tipul** parametrului a carui monitorizare este dorita, **valoarea intervalului de raportare periodica** in milisec. (valoarea 0 indica lipsa raportarii periodice), si **valori ale pragurilor de alertare** (superior si / sau inferior, valoarea 0 indica lipsa pragului).

Efectul este apelul **addParam()** prin care obiectul **Monitor** creeaza si stocheaza un nou parametru (transmitandu-i si referinta de tip **EventsHandler**), si ii stocheaza separat tipul. **Daca primeste valori nenule** pentru interval, **addParam()** creeaza un obiect de tip **Timer** (care extinde **Thread**) pe care il initializeaza cu intervalul primit si cu referinta obiectului de tip **Parameter**, si lanseaza firul de executie al obiectului de tip **Timer**. Constructorul **Parameter()** initializeaza noul obiect cu valorile primite.

Metoda **run()** a obiectelor de tip **Monitor** foloseste apelul **Thread.sleep(period)** pentru a-si suspenda periodic executia. La reluarea executiei se apeleaza metoda **measure()** (a obiectului de tip **MeasuringTool** sau **MeasuringAPI**) furnizandu-i un tablou cu tipurile curent monitorizate, iar cu valorile obtinute se actualizeaza valorile curente ale fiecarui parametru, si apoi se genereaza catre obiectele **Parameter** cereri **checkThresholds()**.

2. Sa se scrie codul metodei **run()** a obiectelor **Monitor**. **(10 pct.)**

Metoda **checkThresholds()** compara valoarea curenta cu pragurile (daca acestea au valori nenule). **Daca unul dintre praguri este depasit**, este creat un obiect **Report** initializat cu **motivul generarii lui** (depasirea pragului superior/ inferior codificata prin 1/ 2), **tipul parametrului** si **valoarea curenta**, si este trimis catre **EventHandler** prin apelul **send()**.

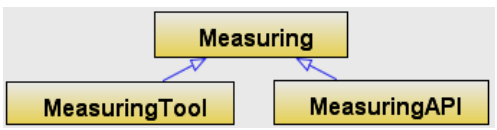
3. Sa se scrie codul metodei **checkThresholds()**. **(10 pct.)**

4. Sa se descrie **sub forma unei diagrame MSC** (continand crearea obiectelor, mesajele schimbate, parametrii acestora) **scenariul** in care este **creat obiectul de tip Monitor**. **(5 pct.)**

5. Sa se descrie **sub forma unei diagrame MSC scenariul** in care sunt **masurati parametrii, verificate pragurile si generat un raport avand ca motivatie depasirea valorii unui prag**. **(5 pct.)**

6. Sa se descrie **sub forma unei diagrame FSM** (masina de stari, **continand stari, tranzitii, evenimente, actiuni, activitati**) **evolutia starilor obiectului** de tip **Monitor** (care trece periodic din starea **asteptare** trecere perioada in starea **masurare si verificare**, ultima formata din substarile **masurare** parametri, **actualizare** valori si **verificare** praguri). **(5 pct.)**

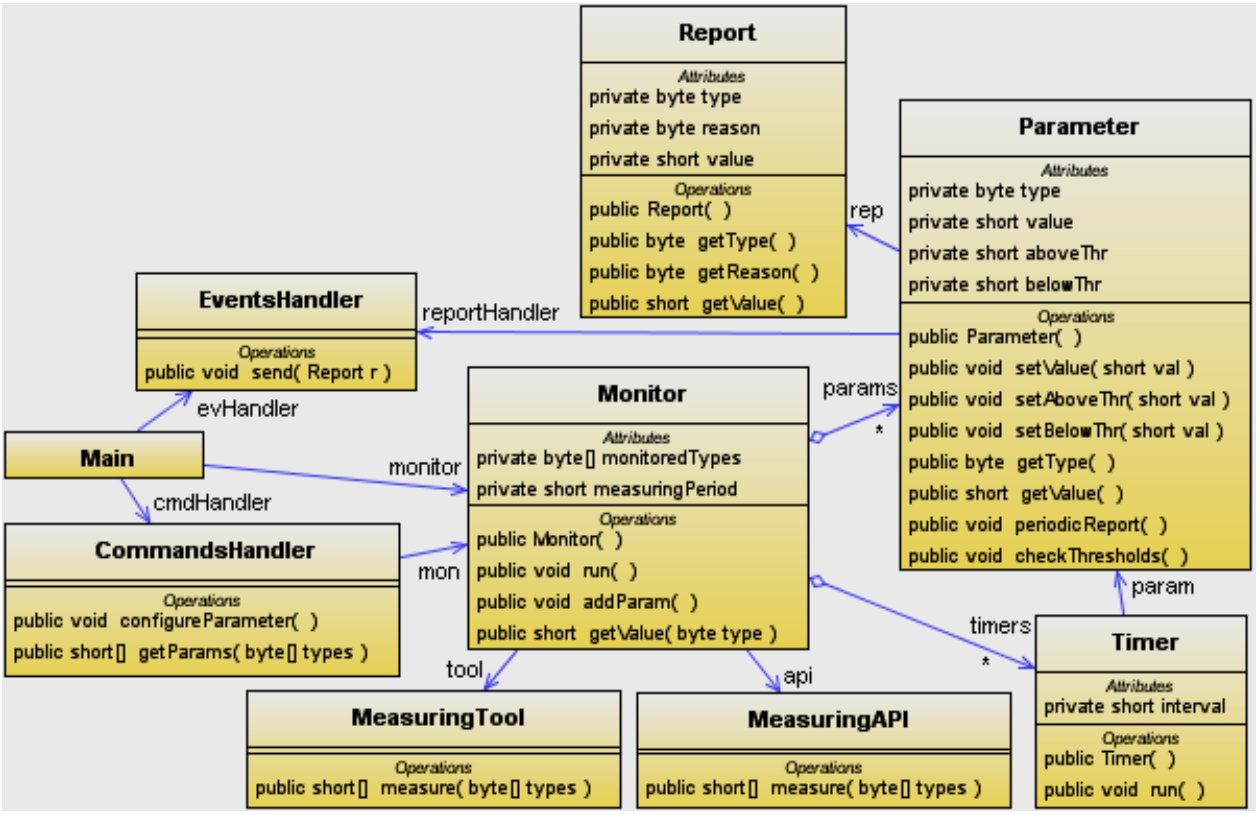
In cazul in care se introduce clasa **Measuring** ca o generalizare a claselor **MeasuringTool** si **MeasuringAPI** (**reconstruite astfel incat sa extinda** clasa **Measuring**):



7. Sa se **alcatuiasca** o **diagrama UML de obiecte**, folosind urmatoarele obiecte si adaugand toate legaturile necesare intre ele: **(5 pct.)**

: MeasuringTool | : Parameter | : Report | : Timer | : EventsHandler | : Main | : Monitor | : Measuring | : Parameter

09.02.2009 **Subiect tip B** Se da **diagrama UML de clase**:



Clasa **Main** creeaza obiectele din clasa **Monitor** (care extinde **Thread**), **EventsHandler** si **CommandsHandler** si face legaturile dintre ele. Constructorul **Monitor()** primeste **perioada efectuarii masuratorilor**, un **flag care indica metoda de masurare** (**Tool** sau **API**) si o **referinta catre un obiect** de tip **EventsHandler** (care va fi transmisa obiectelor de tip **Parameter** si **Report** in momentul crearii acestora), **sa creeze spatiul necesar tratarii a 40 de parametri**, si **sa lanseze firul de executie** al **Monitor**.

Apelul **configureParameter()** primeste **tipul** parametrului a carui monitorizare este dorita, **valoarea intervalului de raportare periodica** in milisec. (valoarea 0 indica lipsa raportarii periodice), si **valori ale pragurilor de alertare** (superior si / sau inferior, valoarea 0 indica lipsa pragului).

Efectul este apelul **addParam()** prin care obiectul **Monitor** creeaza si stocheaza un nou parametru (transmitandu-i si referinta de tip **EventsHandler**), si ii stocheaza separat tipul. **Daca primeste valori nenule** pentru interval, **addParam()** creeaza un obiect de tip **Timer** (care extinde **Thread**) pe care il initializeaza cu intervalul primit si cu referinta obiectului de tip **Parameter**, si lanseaza firul de executie al obiectului de tip **Timer**.

1. Sa se scrie codul metodei **addParam()**. **(10 pct.)**

Apelul **getParams()** primeste **un tablou cu tipurile parametrilor a caror valori sunt dorite** si returneaza **un tablou cu valorile parametrilor respectivi**, facand apel la metoda **getValue()** a obiectului **Monitor**. Apelul **getValue()** primeste **tipul unui parametru a carui valoare este dorita**, cauta si returneaza **valoarea parametrului respectiv**.

2. Sa se scrie codurile metodelor **getParams()** si **getValue()**. **(10 pct.)**

Constructorul **Timer()** primeste pe langa intervalul de raportare si o referinta catre obiectul de tip **Parameter** pe care o stocheaza. Metoda **run()** a obiectelor de tip **Timer** foloseste apelul **Thread.sleep(interval)** pentru a-si suspenda periodic executia si a genera catre obiectul clasei **Parameter** o cerere **periodicReport()**.

Metoda **periodicReport()** creeaza un obiect **Report** pe care il initializeaza cu **motivul generarii lui** (codificat prin valoarea 0), **tipul parametrului** si **valoarea curenta**, si il trimite catre **EventHandler** prin apelul **send()**.

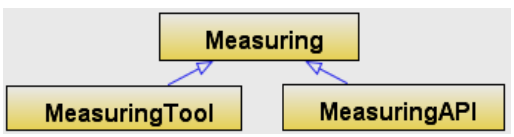
3. Sa se scrie codul metodei **run()** a clasei **Timer** si codul metodei **periodicReport()**. **(10 pct.)**

4. Sa se descrie **sub forma unei diagrame MSC** (continand crearea obiectelor, mesajele schimbate, parametrii acestora) **scenariul** in care **un parametru este monitorizat prin raportare periodica**. **(5 pct.)**

5. Sa se descrie **sub forma unei diagrame MSC scenariul** in care **un nou parametru este creat si configurat**. **(5 pct.)**

6. Sa se descrie **sub forma unei diagrame FSM** (masina de stari, **continand stari, tranzitii, evenimente, actiuni, activitati**) **evolutia starilor obiectului** de tip **Timer** (care initial este **inactiv**, apoi devine **activ**, stare in care periodic trece din substarea **asteptare trecere perioada** in substarea **cerere raport**). **(5 pct.)**

In cazul in care se introduce clasa **Measuring** ca o generalizare a claselor **MeasuringTool** si **MeasuringAPI** (**reconstruite astfel incat sa extinda** clasa **Measuring**):



7. Sa se **alcatuiasca** o **diagrama UML de obiecte**, folosind urmatoarele obiecte si adaugand toate legaturile necesare intre ele: **(5 pct.)**

: MeasuringAPI | : Parameter | : Report | : Timer | : CommandsHandler | : Main | : Monitor | : Measuring | : Parameter