Zwischentest IT 2 WS 2004/2005 am 29.11.2004

Name:
MatrNr:

Beispiel 1 [15 Punkte]

Implementieren Sie die Methode *inform()* in der Klasse *Maschine2*. Verwenden Sie dazu die Klasse *Maschine1* als Vorlage.
Im Gegensatz zu *Maschine1* benötigt *Maschine2* zwei unterschiedliche Typen von Material, die in den Auftragslisten *input1* und *input2* zur Verfügung gestellt werden. Für die Fertigung eines Stückes benötigt *Maschine2* genau 1 Stück von *input1* und 1 Stück von *input2*. Zur Fertigung von *k* Stück werden daher jeweils *k* Stück von *input1* und *input2* benötigt.
Beachten Sie also, dass mit *getAuftrag()* jeweils die gleiche Stückanzahl von *input1* und *input2* angefordert werden muss. Die in einer Auftragsliste vorhandene Stückanzahl kann mit der Methode *getStueckanzahl()* abgerufen werden.

```
/**
 * Maschine mit zwei Arten Material
 */
class Maschine2 implements ZeitlicherProzess
{
    Auftragsliste input1;   // Liste des bereitstehenden Materials 1
    Auftragsliste input2;   // Liste des bereitstehenden Materials 2
    Auftragsliste output;   // Liste, in die die fertig gestellten
                            // Stück eingetragen werden
    int produktionsdauer;   // Von der Maschine für die Produktion
                            // benötigte Zeit
    int wartezeit;          // Wartezeit, wenn keine Produktion
                            // durchgeführt wird
    int maximaleKapazitaet; // Maximale Kapazität der Maschine
    int aktuelleBelegung;   // Aktuelle Belegung der Maschine


    /* Wird aufgerufen, wenn eine Produktion fertig gestellt ist,
     * oder wenn die Wartezeit verstrichen ist.
     */
    void inform()
    {
        …
    }

}


/**
 * Beschreibt eine Liste in die Aufträge hinzugefügt und aus der
 * Aufträge entnommen werden können.
 */
class Auftragsliste
{
    /**
     * Entnimmt Aufträge aus der Auftragsliste.
     * @param maximaleKapazitaet: maximale Stückanzahl, die
     * entnommen werden kann.
     * @return Anzahl der Stücke, die tatsächlich übergeben werden,
     * wenn weniger als maximaleKapazitaet Stück vorhanden sind.
     */
    int getAuftrag(int maximaleKapazitaet);


    /**
     * Fügt anzahlNeueStueck neue Aufträge hinzu.
     */
    void addAuftrag(int anzahlNeueStueck);


    /**
     * Übergibt die aktuelle Stückanzahl in der Liste.
     */
    int getStueckanzahl();
}
```

```
class Clock
{
    /**
     * Dadurch wird prozess nach wartezeit Zeiteinheiten
     * benachrichtigt.
     */
    static void wait(int wartezeit, ZeitlicherProzess prozess);
}


/**
 * Maschine mit einer Art Material
 */
class Maschine1 implements ZeitlicherProzess
{
    Auftragsliste input;    // Liste des bereitstehenden Materials
    Auftragsliste output;   // Liste, in die die fertig gestellten
                            // Stück eingetragen werden
    int produktionsdauer;   // Von der Maschine für die Produktion
                            // benötigte Zeit
    int wartezeit;          // Wartezeit, wenn keine Produktion
                            // durchgeführt wird
    int maximaleKapazitaet; // Maximale Kapazität der Maschine
    int aktuelleBelegung;   // Aktuelle Belegung der Maschine


    /* Wird aufgerufen, wenn eine Produktion fertiggestellt ist,
     * oder wenn die Wartezeit verstrichen ist.
     */
    void inform()
    {
        if(aktuelleBelegung > 0)
        {   // d.h. die Maschine hat produziert
            output.addAuftrag(aktuelleBelegung);
        }

        aktuelleBelegung = input.getAuftrag(maximaleKapazitaet);

        if(aktuelleBelegung > 0)
        {
            Clock.wait(produktionsdauer,this);
        }
        else
        {
            Clock.wait(wartezeit,this);
        }
    }
}
```

Beispiel 2 [15 Punkte]

Markieren Sie, welche der Aussagen über die untenstehenden Deklarationen wahr(W) bzw. falsch(F) sind.

```
class Studium
{
    int studierende;
    int absolventen;

    int getSWS(int semester);
    void inskribiere(int anzahl);
}

class Industrielogistik extends Studium
{
    int computeruebungen;
    Studienplan aktuellerStudienplan;

    String getErstenAbsolventen();
    void evaluiere();
}
```

	W	F	
01.	□	□	*Studium* ist eine Klasse.
02.	□	□	*aktuellerStudienplan* ist eine Klasse.
03.	□	□	*studierende* ist ein Attribut.
04.	□	□	*inskribiere* ist eine Methode.
05.	□	□	*Industrielogistik* ist Unterklasse von *Studium*.
06.	□	□	*computeruebungen* ist ein Attribut von *Industrielogistik*.
07.	□	□	*getErstenAbsolventen* ist eine Methode von *Industrielogistik*.
08.	□	□	Industrielogistik erbt *getSWS*.
09.	□	□	Studium erbt *ersterAbsolvent*.
10.	□	□	*Industrielogistik* erbt die Methode *absolventen*.
11.	□	□	Der Zustand eines Objektes der Klasse *Studium* ist durch die Werte von *studierende* und *absolventen* festgelegt.
12.	□	□	Der Zustand eines Objektes der Klasse *Industrielogistik* ist durch die Werte von *computeruebungen* und *aktuellerStudienplan* festgelegt.
13.	□	□	Das Verhalten eines Objektes der Klasse *Studium* ist durch *getSWS* und *inskribiere* festgelegt.
14.	□	□	Objekte der Klasse *Studium* verstehen die Nachricht *getSWS*.
15.	□	□	Objekte der Klasse *Industrielogistik* verstehen die Nachricht *inskribiere*.