UNIVERSITY OF
WATERLOO

CS247

Software Abstraction and Specification
Midterm Supplemental Sheet

Date: 23-June-2011
Time: 4.30–6.30pm

Spring 2011

5 pages
(including cover sheet)

WATCARD example used in Questions 2-6

Questions 2 through 6 ask you to implement code fragments of a composite class WatCard. Your answers to these questions should agree with one another (e.g., they should compile, link, and work together).

Much of the textual description of this problem comes from University of Waterloo Web pages about the WatCard and about Meal Plans.

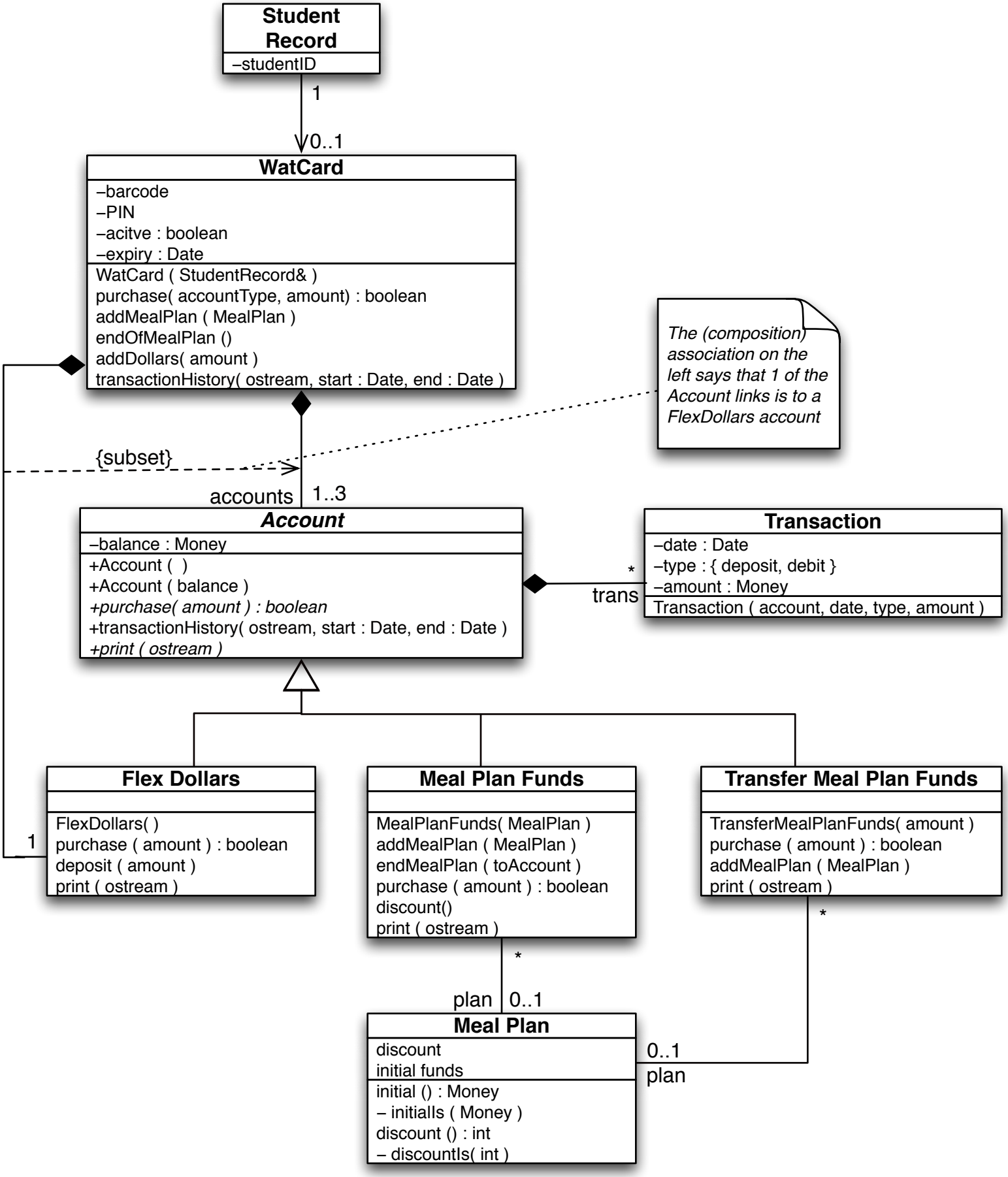
The WatCard has multiple uses, including

Debit Card: The WatCard is a debit card for the student's Flex Dollars account. Money can be added to the account at any time. Purchases paid for with Flex Dollars are subject to HST (a provincial 13% sales tax).

Meal Plan Card: If the student buys a meal plan for the term, then an initial amount of funds are put into the student's Meal Plan Funds account and the WatCard acts as a debit card for this account. Most meal plans offer discounts on all such purchases. Also, meal plan purchases are tax exempt (i.e., they are not subject to HST). If the Meal Plan Funds account runs out of funds, then food purchases are automatically charged to the Flex Dollars account, which means that no discount applies and there is HST.

Expired Meal Plan Card: If there is money left in a student's Meal Plan Funds account at the end of a term, those funds are transferred to the student's Transfer Meal Plan Funds (TMPF) account and the WatCard acts as a debit card for this account. If the student then purchases a meal plan for the current term, purchases using the TMPF account take advantage of that meal plan's discount. But these purchases are subject to HST. If the TMPF account runs out of funds, then food purchases are automatically charged to the Flex Dollars account, which means that no discount applies.

WATCARD UML model used in Questions 2-6



WATCARD C++ class declaration used in Questions 2-6

```
#include <ostream>
#include "Account.h"
#include "Money.h"
#include "Date.h"

enum AccountType { FLEX, MEAL, TRANSFER };

class WatCard {
public:
     WatCard( const StudentRecord& );                    // constructor
     bool purchase( AccountType, const Money &amount);   // debit amount from specified account; return success code
     void addMealPlan( const MealPlan& );                // add meal plan to WatCard
     void endOfMealPlan();                               // remove meal plan; transfer unused funds to Transfer Meal Plan account
     void addDollars( const Money &amount );             // add money to FlexDollars account
     void transactionHistory( std::ostream &sout, const Date &start, const Date &end ) const;
                                                         // stream transactions for all accounts between start and end dates

private:
     static const int NUM = 3;
     StudentRecord &sr_;
     long barcode_;
     int* PIN_;
     bool active_;
     Date expiry_;
     Account* accounts_[NUM];  // Assume that AccountType (above) indices denote type of Account
                               // (i.e., accounts_[FLEX] is a Flex Dollar acccount
     long newBarcode();
};
```

DATE C++ class declaration; may be useful in Questions 2-6

```
#include <string>
#include <iostream>

class Date {
public:
    Date();                                          // returns new Date = today
    Date (int day, std::string month, int year);     // constructor
    int day() const;                                 // accessor
    std::string month() const;                       // accessor
    int year() const;                                // accessor
    Date incDays (long) const;              // increments Date by some number of days
    Date incMonths (int) const;             // increments Date by some number of months
    Date incYears (int) const;              // increments Date by some number of years
};

bool operator== (const Date&, const Date&);
bool operator!= (const Date&, const Date&);
bool operator< (const Date&, const Date&);
bool operator<= (const Date&, const Date&);
bool operator> (const Date&, const Date&);
bool operator>= (const Date&, const Date&);

std::ostream& operator<< (std::ostream&, const Date&);
std::istream& operator>> (std::istream&, Date&);
```

MONEY C++ class declaration; may be useful in Questions 2-6

```
#include <string>
#include <iostream>

class Money {
public:
    explicit Money( long dollars = 0, long cents = 0);  // Constructor with specified money value
    Money dollars() const;                              // Accessor - dollar value
    Money cents() const;                                // Accessor - cents value
    Money operator+ (const Money& n) const;             // Add two money values; return result
    Money operator- (const Money& n) const;             // Subtract n; return result
    Money operator* (int f) const;                      // Multiply by f%; return result
};

bool operator== (const Money& m, const Money &n);
bool operator!= (const Money& m, const Money &n);
bool operator<  (const Money& m, const Money &n);
bool operator<= (const Money& m, const Money &n);
bool operator>  (const Money& m, const Money &n);
bool operator>= (const Money& m, const Money &n);

std::istream& operator>> (std::istream&, Money&);
std::ostream& operator<< (std::ostream&, const Money&);
```