Name ____

Student number __________________________________

SE 2AA4 Winter 2007

Midterm Test Answer Key

Instructor: William M. Farmer

Revised: 8 March 2007

You have 50 minutes to complete this test consisting of 6 pages and 15 questions. You may use your notes and textbooks, but you may not use any calculators or other electronic devices. Circle the *best* answer for the multiple choice questions, and write the answer in the space provided for the other questions. Good luck!

(1) [4 pts.] Software engineers have developed effective standardized ways of measuring software quality. Is this statement true or false?

(a) True.

(b) $\boxed{\text{False.}}$

(2) [4 pts.] The C programming language, with its header files, provides better support for modules than does the Java programming language. Is this statement true or false?

(a) True.

(b) $\boxed{\text{False.}}$

(3) [4 pts.] A Java class can implement no more than one Java interface. Is this statement true or false?

(a) True.

(b) $\boxed{\text{False.}}$

(4) [4 pts.] It is plausible that increasing reusability will increase production cost in the short term but decrease it in the long term? Is this statement true or false?

(a) $\boxed{\text{True.}}$

(b) False.

(5) [4 pts.] C source code is normally

(a) Interpreted.

(b) $\boxed{\text{Compiled into machine code and then executed.}}$

(c) Compiled into byte code and then interpreted.

(d) Compiled into byte code, then compiled into machine code, and finally executed.

(6) [4 pts.] Which statement is not true about abstraction?

(a) Abstraction is the opposite of refinement.

(b) Abstraction is a good way of finding a general solution to a problem.

(c) $\boxed{\text{Abstraction is an important component of compilation.}}$

(d) Abstraction is an example of separation of concerns.

(7) [4 pts.] A module's interface should always

(a) Have as few components as possible.

(b) Be hidden from other modules.

(c) Contain data and procedures.

(d) $\boxed{\text{Reveal as little of the module's implementation as possible.}}$

(8) [4 pts.] Which of the following statements defines a macro in C?

(a) `static const int START = -17;`

(b) `typedef float Macro;`

(c) $\boxed{\texttt{\#define START -17}}$

(d) `#include <stdbool.h>`

(9) [4 pts.] In which of the following software development phases is the client usually most involved?

(a) $\boxed{\text{Requirements.}}$

(b) Design.

(c) Implementation.

(d) Verification.

(10) [4 pts.] If the privileges of a unix file are `-rwxrw---x`,

(a) Anyone can read the file.

(b) $\boxed{\text{Anyone can execute the file.}}$

(c) Only the owner of the file can write the file.

(d) When executed the file runs under the privileges of its owner.

(11) [4 pts.] A software product that does not satisfy its requirements can still be

(a) Usable.
(b) Robust.
(c) Reliable.
(d) $\boxed{\text{All of the above.}}$

(12) [4 pts.] Suppose that a software module M has an interface I of services that are provided to a set of other software modules. Which change to M would *not* require any changes to the other modules?

(a) Removing a component of I.
(b) $\boxed{\text{Modifying the implementation of a component of } I.}$
(c) Modifying the specification of a component of I.
(d) None of the above.

(13) [4 pts.] In 2–3 sentences explain why the development of a "little language" is a useful technique for dealing with change.

Answer: Suppose that we have a software design problem whose requirements are expected to moderately change over time. The idea behind the little languages technique is to develop a special-purpose language of primitive components that can be assembled to produce solutions to the software design problem for many different, but closely related, requirements. Thus each time the problem's requirements change, the primitives of the language can be reassemble to form a new solution. In this way, the language solves a family of related requirements.

(14) [12 pts.] Recall that the interface of the `VectorAdt` module of Exercise 3 contains the following C procedures:

(a) `float get_x(Vector v);`
(b) `float get_y(Vector v);`
(c) `Vector i_vector();`
(d) `Vector j_vector();`
(e) `Vector mul(float r, Vector v);`
(f) `Vector add(Vector u, Vector v);`

The *dot product* of two vectors $V_1 = (a_1, b_1)$ and $V_2 = (a_2, b_2)$ is the real number $V_1 \bullet V_2 = a_1 \cdot a_2 + b_1 \cdot b_2$. Write a procedure

```
float dot(Vector u, Vector v);
```

in C to be added to the `VectorAdtPlus` module that implements the dot product of two vectors. Assume that the procedure `dot` has access to the interface of `VectorAdt` but not to the implementation of `VectorAdt`.

Answer:

```
float dot(Vector u, Vector v) {
  return (get_x(u) * get_x(v)) + (get_y(u) * get_y(v));
}
```

(15) Let O be an object that instantiates the following Java class. O is can be viewed as software module that implements an "account" data structure.

```
public class Account {

    private int balance_ = 0;

    public int value() {
        return balance_;
    }

    public void reset() {
        balance_ = 0;
    }

    public void add(int x) throws NegativeBalanceException {
        balance_ = balance_ + x;
        if (isNegative()) {
            throw new NegativeBalanceException(balance_);
        }
    }

    private boolean isNegative() {
        return balance_ < 0;
    }

}
```

(a) [6 pts.] What are the components of the interface of O?

Answer: `value, reset, add`.

(b) [6 pts.] Which of these components are selectors for the data structure implemented by O?

Answer: `value`.

(c) [6 pts.] Which of these components are mutators for the data structure implemented by O?

Answer: `reset, add`.

(d) [6 pts.] What condition will cause `NegativeBalanceException` to be thrown?

Answer: The `NegativeBalanceException` is thrown whenever the value of `balance_` is negative after it is changed by the `add` procedure.

(e) [12 pts.] Write a Java method

```
public int set(int x);
```

that changes the balance of O to x and returns what the balance of O was before this change was made. Assume that this method will be put into a definitional extension of `Account`.

Answer:

```
public float set(int x) {
    int a = value();
    try {
        add(x - a);
    }
    catch (NegativeBalanceException e) {
        System.out.println(e.toString());
    }
    return a;
}
```