ECE 30862 Fall 2013, First Exam, KEY

DO NOT START WORKING ON THIS UNTIL TOLD TO DO SO. LEAVE IT ON THE DESK.

You have until 7:25 to take this exam.

Your exam should have 9 pages total (including this cover sheet). *Please let Prof. Midkiff know immediately if it does not.*

This exam is open book, open notes, but absolutely no electronics. If you have a question, please ask for clarification. If the question is not resolved, state on the test whatever assumptions you need to make to answer the question, and answer it under those assumptions. *Check the front board occasionally for corrections.*

I have neither given nor received help during this exam from any other person or electronic source, and I understand that if I have I will be guilty of cheating and will fail the exam and perhaps the course.

Name (must be signed to be graded):

Name (printed, worth 1 pt):

Last four digits of your ID:

If you have a question about an answer, think an answer is wrong, or don't think the question was clear and/or correct, send me email

Prof. Midkiff

Question 1 (Java). Write what is printed beside each function call to m1, m2 or m3 in `main` in the following program (1.9pts each).

```
import java.io.*;
class B {

   public double f;

   public B( ) {
      f = -1;
   }

   public void m1(double ff) {
      f = ff;
      System.out.println("B's");
   }

   public void m2( ) {
      System.out.println("B's");
   }

   public void m3(double ff) {
      f = ff;
      System.out.println("B's");
   }
}
import java.io.*;

public class D extends B {

   public int i;

   public D( ) {
      f = -1.0;
   }

   public void m2( ) {
      System.out.println("D's");
   }

   public void m3(int ii) {
      i = ii;
      System.out.println("D's");
   }
}
```

```
class P1 {

   public static void main(String args[]) {
      B b1 = new B( );
      D d1 = new D( );
      B b2 = null;

      System.out.println("b1");

      b1.m1(1); // B's

      b1.m2( ); // B's

      System.out.println("d1");

      d1.m1(1.0); // B's

      d1.m2( ); // D's

      d1.m3(3); // D's

      d1.m3(3.0); // B's

      System.out.println("b2");

      b2 = (B) d1;

      b2.m1(1.0); // B's

      b2.m3(3); // B's.  This is an interesting
      // case.  Because the reference is of type
      // B the // class B is looked at to see
      // what matches.  A conversion from int
      // to float lets m3(double) be matched.
      // The call is then made through the VFT
      // for m3(double), which is B's.  Note
      // that this is not hiding.  If Java had
      // hiding like C++'s, the call to
      // d1.m2(3.0) above would call D's m3(int).

      b2.m3(3.0); // B's
   }
}
```

Question 2 (C++). Write what is printed beside each function call to m1, m2 or m3 in `main` in the following program (1.9 pts each).

```
#include <iostream>
#include <string>
using namespace std;

class B {

private:
   double f;

public:
   B( ) { f = -1; }
   ~B( ) { }

   virtual void m1(double ff) {
      f = ff;
      cout << "B's" << endl;
   }

   virtual void m2( ) {
      cout << "B's" << endl;
   }

   virtual void m3(double ff) {
      f = ff;
      cout << "B's" << endl;
   }
};

class D : public B {
private:
   int i;
public:
   D( ) {
      i = -1;
   }

   ~D( ) { }

   virtual void m2( ) {
      cout << "D's" << endl;
   }

   virtual void m3(int ii) {
      i = ii;
      cout << "D's" << endl;
   }
};
```

```
int main(int argc, char * argv[ ]) {

   B* b1 = new B( );
   D* d1 = new D( );
   B* b2;

   b1->m1(1); // B's

   d1->m2( ); // D's

   d1->m3(3); // D's

   d1->m3(3.0); // D's

   b2 = (B*) d1;

   b2->m2( ); // D's

   b2->m3(3); // B's

   b2->m3(3.0); // B's
}
```

Question 3 (C++). Write what is printed beside each function call to m1, m2 or m3 in `main` in the following program. The only significant differences between this program and hte program of Question 2 is that this program does **not** have *virtual* functions and the program of Question 2 does. (1.9 pts each).

```
#include <iostream>
#include <string>
using namespace std;
class B {
private:
   double f;
public:
   B( ) { f = -1; }
   ~B( ) {  }

   void m1(double ff) {
      f = ff;
      cout << "B's" << endl;
   }

   void m2( ) {
      cout << "B's" << endl;
   }

   void m3(double ff) {
      f = ff;
      cout << "B's" << endl;
   }
};

class D : public B {
private:
   int i;
public:
   D( ) {
      i = -1;
   }

   ~D( ) { }

   void m2( ) {
      cout << "D's" << endl;
   }

   void m3(int ii) {
      i = ii;
      cout << "D's" << endl;
   }
};
```

```
int main(int argc, char * argv[ ]) {

   B* b1 = new B( );
   D* d1 = new D( );
   B* b2;

   b1->m1(1); // B's

   d1->m1(1.0); // B's

   d1->m2( ); // D's

   d1->m3(3); // D's

   d1->m3(3.0); // D's

   b2 = d1;

   b2->m2( ); // B's

   b2->m3(3); // B's

   b2->m3(3.0); // B's
}
```

Question 4 (C++). Write what is printed beside each function call to m1, m2 and m3 in `main` in the following program. The significant differences between this program and the program of Question 2 is that object variables rather than pointers to objects are used to access functions. Like Question 2 the functions are virtual. (1.9 pts each).

```
#include <iostream>
#include <string>
using namespace std;

class B {

public:
   B( ) { f = -1; }
   ~B( ) { }

   virtual void m1(double ff) {
      f = ff;
      cout << "B's" << endl;
   }

   virtual void m2( ) {
      cout << "B's" << endl;
   }

   virtual void m3(double ff) {
      f = ff;
      cout << "B's" << endl;
   }

   private:
      double f;
};

class D : public B {
public:
   D( ) {
      i = -1;
   }

   ~D( ) { }

   virtual void m2( ) {
      cout << "D's" << endl;
   }

   virtual void m3(int ii) {
      i = ii;
      cout << "D's" << endl;
   }
private:
   int i;
};
```

```
int main(int argc, char * argv[ ]) {

   B b1; // zero arg constructor
   D d1; // zero arg constructor
   B b2; // zero arg constructor

   b2 = (B) d1;
   b2.m1(1); // B's
   b2.m1(1.0); // B's
   b2.m2( ); // B's
   b2.m3(3); // B's
   b2.m3(3.0); // B's
}
```

Question 5 (Java).
Answer the true false questions below (2 pts each).

```
interface I1 {
   abstract void m1( );
   abstract void m1(int i);
}

interface I2 {
   abstract void m1(int i);
   abstract void m3( );
}

abstract class B {
   abstract void m1( );

   void m5(int i) {
      System.out.println("i is "+i);
   }
}
```

```
public class D extends B implements I1, I2 {
   public D( ) { }

   public void m1( ) {
      System.out.println("m1 in D");
   }

   public void m1(int i) {
      System.out.println("m1(int) in D");
   }

   public void m3( ) {
      System.out.println("m3 in D");
   }
}

class P5 {
   public static void main(String args[]) {
      D d = new D( );
      d.m1( );
   }
}
```

Answer the following questions as true or false.

- Abstract classes can implement both abstract and non-abstract methods. **true**
- There will be an error because multiple interfaces are implemented. **false**
- There will be an error because multiple interfaces specify the same method. **false**
- Interfaces and abstract classes cannot be used together. **false**
- The definition of `void m1( )` in `class D` fulfills the requirement that there be an definition of the method in both abstract class B and interface I1. **true**

Question 6 (C++). (5 points)
Given the code below:

```
#include <iostream>
#include <string>
using namespace std;

class B {
private:
   double f;
public:
   B( ) { f = -1; }
   ~B( ) { }

   virtual void m1(double ff) {
      f = ff;
      cout << "B's" << endl;
   }

   virtual void m2( ) = 0;
};
```

```
class D : public B {
private:
   int i;
public:
   D( ) {
      i = -1;
   }

   ~D( ) { }

   virtual void m3(int ii) {
      i = ii;
      cout << "D's" << endl;
   }
};

int main(int argc, char * argv[ ]) {
   D* d = new D( );
}
```

that gets the following error message when compiled:

```
// p2.cpp: In function int main(int, char**):
// p2.cpp:39: error: cannot allocate an object of abstract type D
// p2.cpp:22: note:   because the following virtual functions are pure within D:
// p2.cpp:16: note:  virtual void B::m2()
```

explain in 20 words or less why.

Something along the lines of:
Tried to create an object from an abstract class, which is illegal.

Question 7 (C++).
In function `main`, circle the legal accesses, both members and functions (1.9 pts each).

```
#include <iostream>
#include <string>
using namespace std;

class B {
private:
   double v;
public:
   B( ) { v = -1; }
   ~B( ) { }

   virtual void m1(double vv) {
      v = vv;
   }

   virtual double getV( ) { return v;}
};

class D : private B {
private:
   int i;
public:
   int j;

   D( ) {
      i = -1;
   }

   ~D( ) { }

   virtual void m2( ) {
      cout << "D's" << endl;
   }
};
```

```
int main(int argc, char * argv[ ]) {

   B* b = new B( );
   D* d = new D( );
   double f;

   b->m1(1.0); // {\bf legal}

   f = b->getV( ); // {\bf legal}

   b->v = 1.0;

   d->m1(1.0);

   f = d->getV( );

   f = d->v;

   d->m2( ); // {\bf legal}

   d->j = 1; // {\bf legal}

   d->i = 1;
}
```

Question 8 (Both C++ and Java, 2pts each).
If they give a container that is ok other than the one I mention, count it as correct.
Give a container class that is well-suited for each application below:

- Many inserts into the front and back of the container
  A list.

- Need to lookup objects in the container using a unique key.
  a map or a vector. A map is better since a vector needs nearly contiguous keys to be practical.

- Objects will be inserted at the front and removed from the back.
  A queue.

- Objects should only appear once in the container even if inserted many times.

A set.

Question 9 (2 pts each). Give short (one to five word answers) for each question below on a variety of topics.

- What C++ keyword used in a class allows a specific function or class to access the private functions and fields of a class or object?
  friend.

- Who can access *protected* fields or functions in a C++ class?
  A derived class, an inheriting class.

- true or false: there is a copy of a `static` field of a Java or C++ class in each object instance of that class.
  false