# Problem 1

**1. Which of the B.m methods below are function substypes of A.m? For each of the B.m methods answer whether the method would overload or override A.m in Java.**

class A {

Object m(X y, String s);

}

class B extends A {

X m(X y, String s);

Y m(Object y, Object s);

Object m(X y, String s) throws RuntimeException;

Z m(Y y, String s);

}

The function subtypes are:

X m(X y, String s);

Y m(Object y, Object s);

For each of the B.m methods:

X m(X y, String s); -- Override

Y m(Object y, Object s); -- Overload

Object m(X y, String s); -- Neither

Z m(Y y, String s); -- Overload

**2. For each pair of specifications below, answer whether the extending class is a true subtype of its superclass. Explain your answer.**

class Triangle {

// modifies: this

// effects: this\_post.a=a, this\_post.b=b, this\_post.c=c

void setSides(int a, int b, int c);

}

class IsoscelesTriangle extends Triangle {

// modifies: this

// effects: this\_post.a=a, this\_post.b=b, this\_post.c=b

void setSides(int a, int b, int c);

}

Isosceles triangle is a true subtype – the constructor in isosceles triangle is stronger because it checks to see if two sides are equal. I thus has a stronger spec and can replace the Triangle class anywhere.

abstract class Vertebrate extends Animal {

// returns: an integer > 0

int neckBones();

}

class Squid extends Vertebrate {

// returns: 0 neck bones

int neckBones();

}

class Human extends Vertebrate {

// returns: 7 neck bones

int neckBones();

}

Squid is not a true subtype. It doesn’t return a number of neckbones greater than 0, so it cannot be substituted in for a vertebrate object.

Human is a true subtype. Human has a stronger specification because the neckbones function returns 7 which is greater than 0. It can be substituted in for vertebrate.

class Bicycle {

int cadence;

int speed;

int gear;

// effects: creates a new Bicycle

Bicycle(int startCadence, int startSpeed, int startGear);

// modifies: this

// effects: this\_post.cadence=newCadence, this\_post.speed=newSpeed, this\_post.gear=newGear

void setParameters(int newCadence, int newSpeed, int newGear);

}

class MountainBike extends Bicycle {

int height;

// effects: creates a new MountainBike

MountainBike(int startCadence, int startSpeed, int startGear, int startHeight);

// modifies: this

// effects: this\_post.height=newHeight

void setHeight(int newHeight);

}

Mountainbike is not a true subtype because it’s constructor expects the startHeight argument – it cannot be substituted in for the bicycle class.

class Account {

// modifies: this

// effects: this\_post.amount = this\_pre.amount + d;

void deposit(int d);

}

/\* An account that works safely when multiple transactions attempt deposits simultaneously \*/

class ConcurrentAccount extends Account {

// modifies: this

// effects: this\_post.amount = this\_pre.amount + d;

// throws: AbortException if another transaction is in the process of depositing money

void deposit(int d);

}

A concurrent account is not a true subtype because it throws an exception when some additional criteria is met – if another transaction is in the process of depositing money. This means it cannot replace the normal account class in every instance.

# Reflection

**1. In retrospect, what could you have done better to reduce the time you spent solving this assignment?**

I should have spent more time optimizing the design of the graph class from hw4 so I wouldn’t have had to redo too much code.

**2. What could have we done better to improve your learning experience in this assignment?**

Maybe done a bit more explicit comparison of the running time differences between various graph implementations to give us a better idea of how we should be designing the class.

**3. What do you know now that you wish you had known before beginning the assignment?**

The true extent of the hw5 test cases so we’d be more concerned with efficiency.

# Collaboration

I did not collaborate with other students

# Changes

I switched from an adjacency matrix to an adjacency list to cut down of memory usage and running time.