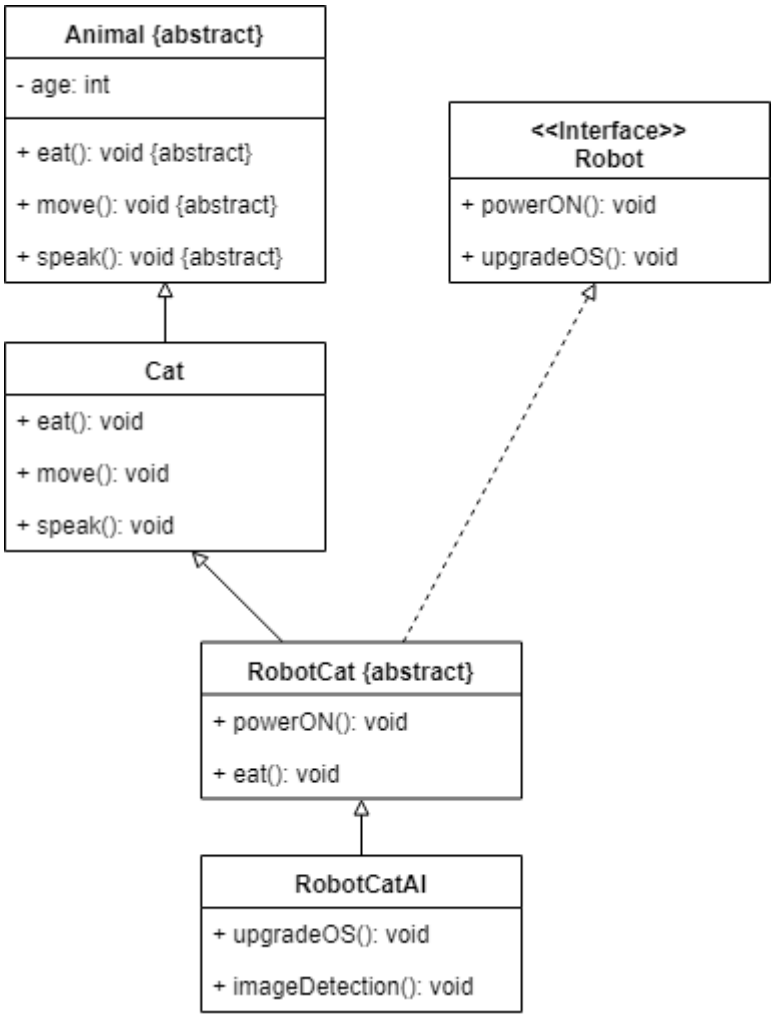
20th CSEC – Past Year Paper Solution *2018-2019 Sem 2*
CE/CZ 2002 – Object Oriented Design & Programming

1)

a)

i)



ii)

```
public abstract class Animal {
   private int age;
   public abstract void eat();
   public abstract void move();
   public abstract void speak();
}

public class Cat extends Animal {
   public void eat() {
         System.out.println("eat");
   }
   public void move() {
         System.out.println("move");
   }
   public void speak() {
         System.out.println("speak");
```

```
    }
}

public abstract class RobotCat extends Cat implements Robot {
    public void powerON() {
        System.out.println("powerON");
    }
    public void eat() {
        System.out.println("eat");
    }
}

public class RobotCatAI extends RobotCat {
    public void upgradeOS() {
        System.out.println("upgradeOS");
    }
    public void imageDetection() {
        System.out.println("imageDetection");
    }
}
```

b)

Line	Output	Reason
1	AD	The method fun(double) in A matches the argument type.
2	AD	The method fun(double) in A matches the argument type.
3	BO	The method fun(Object) in B best matches the argument type, as Object is a superclass of String.
4	DO	The method fun(Object) in D best matches the argument type, as Object is a superclass of String.

Note: Test is in package p1.s1 as D is a class with default access modifier. The method fun(String) in A is not accessible in Test due to the access specifier of the method.

2)

a)

Line	Output	Reason
1	C	The method fun(int) in C matches the argument type.
2	B	The method fun(int) in B (superclass) best matches the argument type.
3	A	The only method in A is fun(double). The argument is promoted from int to double, which matches the argument type.
4	B	The only method in B is fun(int), which matches the argument type.
5	F	The superclass E overrides fun(int) of the superclass B. At runtime, z's actual type is determined, which is E, hence E's fun(int) is called. (dynamic binding)

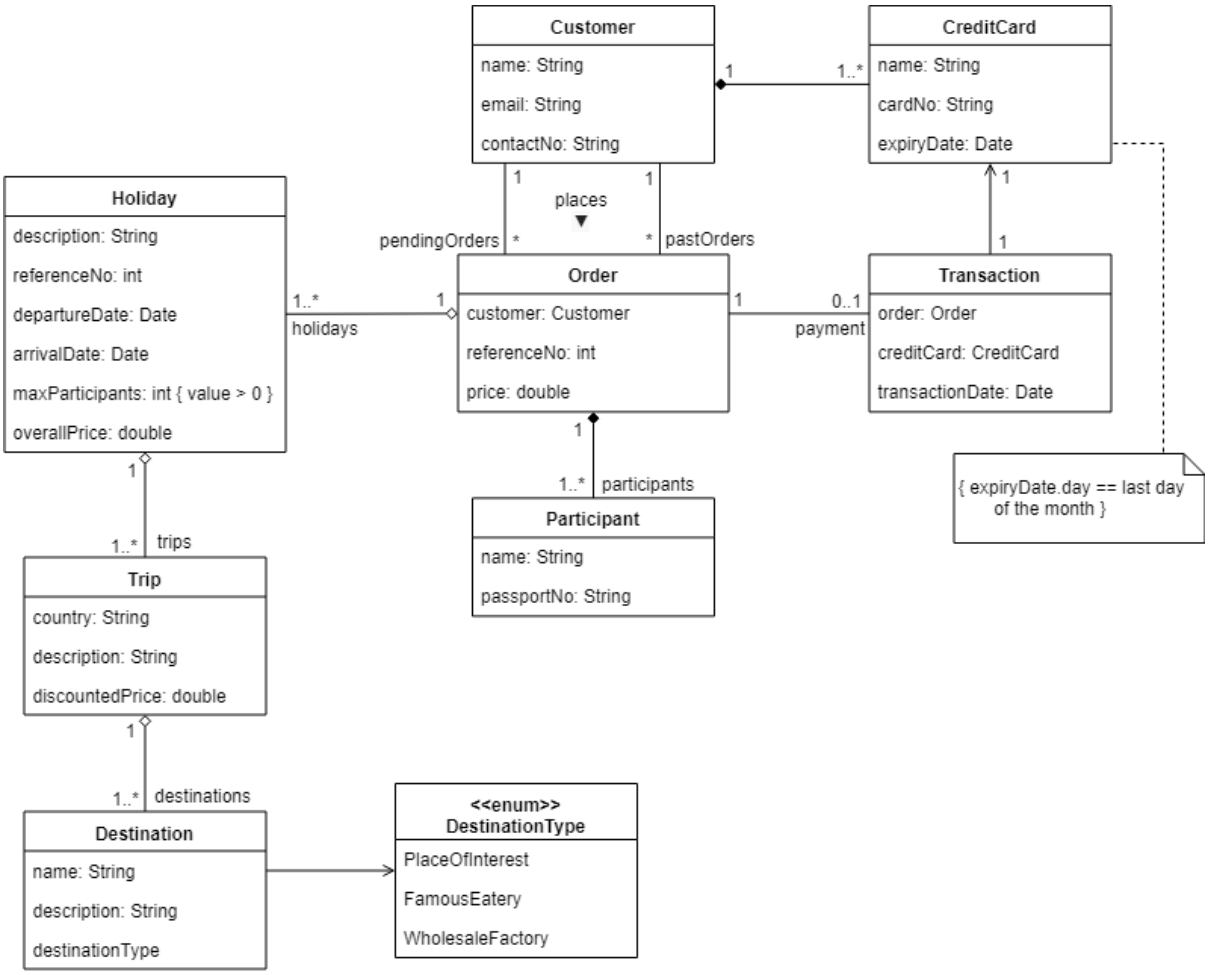
b)

Line	Output	Reason
1	10 0	The no-argument constructor of AA calls print(), which prints the value of a, then the value of b.

2	10 0 A1	The constructor of A1 does not explicitly invoke a superclass constructor, hence the no-argument constructor of the superclass AA will be invoked before the first line. The no-argument constructor of AA calls print(), which prints the value of a, then the value of b. After calling the constructor of the superclass, the codes in the constructor of A1 will run, hence printing "A1".
3	A2 print 0 A2	The subclass A2 overrides print() of the superclass AA. When A2's superclass constructor calls print() at runtime, a2's actual type is determined, which is A2, hence A2's print() is called, which prints "A2 print", then the value of b. After calling the constructor of the superclass, the codes in the constructor of A1 will run, hence printing "A2".
4	A2 print 1	The subclass A2 overrides print() of the superclass AA. At runtime, a2's actual type is determined, which is A2, hence A2's print() is called, which prints "A2 print", then the value of b. The value of b was previously incremented in the 3rd line.
5	10 1	The method print() in AA is called, which prints the value of a, then the value of b.

3)

a)



b)
```
public class Runner {
      private Configurator cfr;

      private Runner(Configurator cfr) {
            this.cfr = cfr;
      }

      public int startTest() {
            int status = start();
            return status;
      }

      private int start() {
            int status = -1;
            boolean result = validate();
            if (!result)          // result == false
                  return 0;       // status = 0
            double valA = cfr.getConfigA();
            if (valA > 99) {
                  Runner r2 = new Runner(cfr);
                  status = r2.startTest();
            }
            return status
      }

      private boolean validate() {
            boolean result = cfr.check();
            return result;
      }
}
```

4)

a)

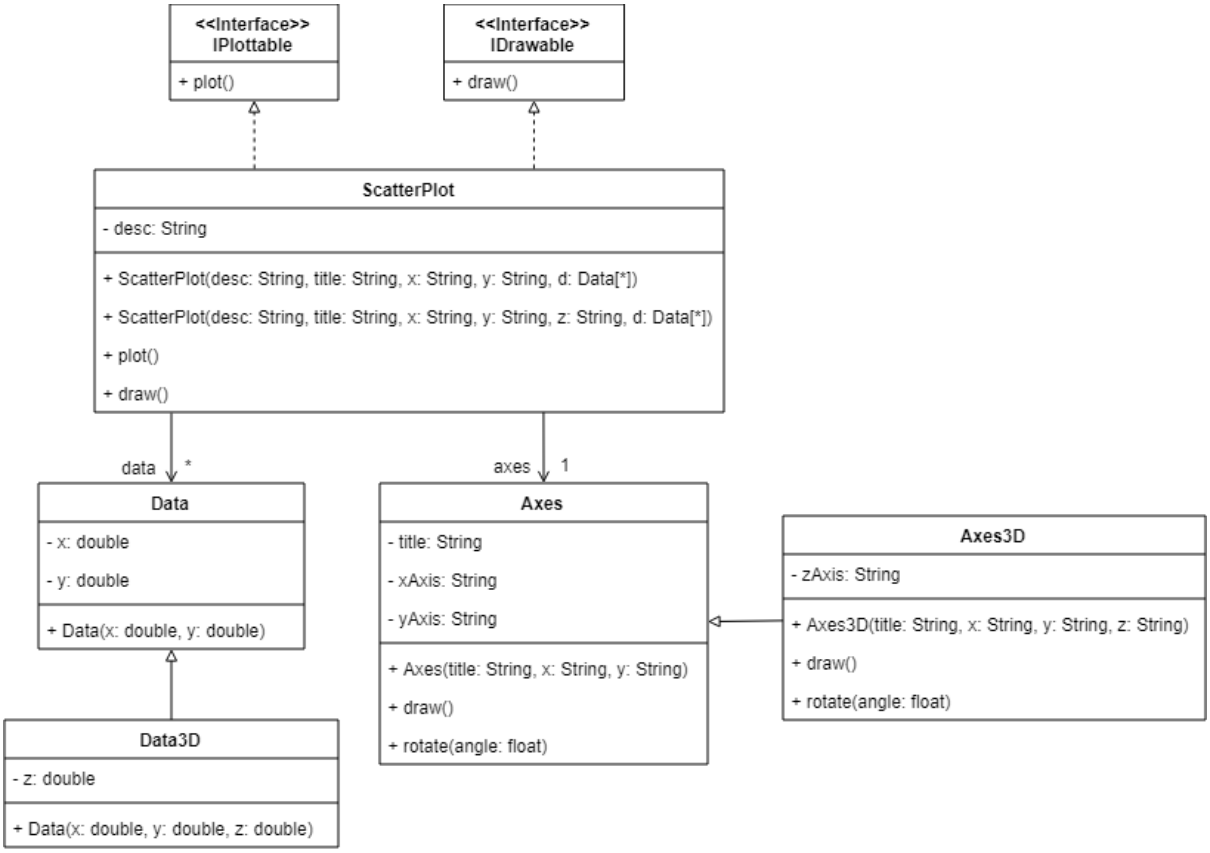
i)
```
// IDrawable.h
#ifndef IDRAWABLE_H
#define IDRAWABLE_H

class IDrawable {
public:
   virtual void draw() = 0;
};

#endif  // IDRAWABLE_H
```

ii) Assume: `IPlottable` class is in a fine named "IPlottable.h"

```
// scatterplot.h
#ifndef SCATTERPLOT_H
#define SCATTERPLOT_H
```

```
#include <string>
#include <vector>
#include "IPlottable.h"
#include "IDrawable.h"
#include "Data.h"

class ScatterPlot: public IPlottable, public IDrawable {
   std::string desc;
   std::vector<Data> data;
   Axes axes;

public:
   ScatterPlot(std::string desc, std::string title,
std::string x, std::string y, const std::vector<Data>& data)
   : desc(desc), axes(title, x, y), data(data) {}
   ~ScatterPlot() {}
   virtual void plot();
   virtual void draw();
};

#endif // SCATTERPLOT_H

// scatterplot.cpp
#include <iostream>
#include "scatterplot.h"

using std::cout;
using std::endl;

void ScatterPlot::plot() {
   cout << "plotting…" << endl;
}

void ScatterPlot::draw() {
   axes.draw();
   cout << "drawing…" << endl;
}
```

b)



Open-Closed Principle: The Axes and Data classes are closed for modification, but open for extension. The Axes class had to be modified to add `rotate()` as it is a behavior required of the current two axes.

Liskov Substitution Principle: The Data3D and Axes3D classes are substitutable with their respective parent classes. That is, the ScatterPlot class can use these classes via pointers of their parent classes, regardless of the actual object type.

*If a z variable was added to the Data/Axes class, then a Data2D/Axes2D subclass extends from it, LSP would be violated (The parent class, Data/Axes, would require more information than its subclass).

ISP/DIP: Creating interfaces to abstract the behavior of the existing Data/Axes classes is a less relevant solution for the question. 3D is more of an extension of Data and Axes than sharing some behavior that could be abstracted. In addition, the existing classes would be impacted by having to make use of the new interface (realization & association).

SRP: Not very relevant to the question.

--End of Answers--

Solver: Foo Jing Ting