Langara College
CPSC 1181 MIDTERM 2
Spring 2017; Fri, March 17th

Jeremy Hilliker

Student Name & ID number: ____

Signature: ____

Length: 11 pages (including cover) - 120 minutes

Instructions:

- Do not open the exam until instructed to do so.
- Ensure that your monitor is turned off, and that the keyboard and mouse are out of the way.
- Place all personal belongings below your desk or at the front of the room.
    - Turn off and put away all electronic devices.
    - Ask permission before retrieving any items from below your desk during the midterm.
- Your answers to this exam must be entirely your own.
    - Strictly no speaking to other students during the midterm.
- Record your answers in the exam answer booklet.
    - Print neatly and legibly. Illegible answers will be marked as incorrect.
    - Use backs of pages only if necessary, and indicate that they are your answer in the regular space.
- The exam is over when announced by the invigilator. Continuing to write may result in a penalty.
- Put your name and ID number on this cover page, and sign.
- Put your name and ID, as well as the midterm information, on the cover of the exam answer booklet.
- When finished, place these question pages inside of your answer booklet, and hand them both to the invigilator.

Q1	10+2	
Q2	16	
Q3	30+2	
Q4	10	
Q5	22+1	
Q6	20+1	
Q7	12	
Q8	0+3	
Total	120+9	

Permitted aids:

- One double-sided letter/A4 size reference sheet of any origin.

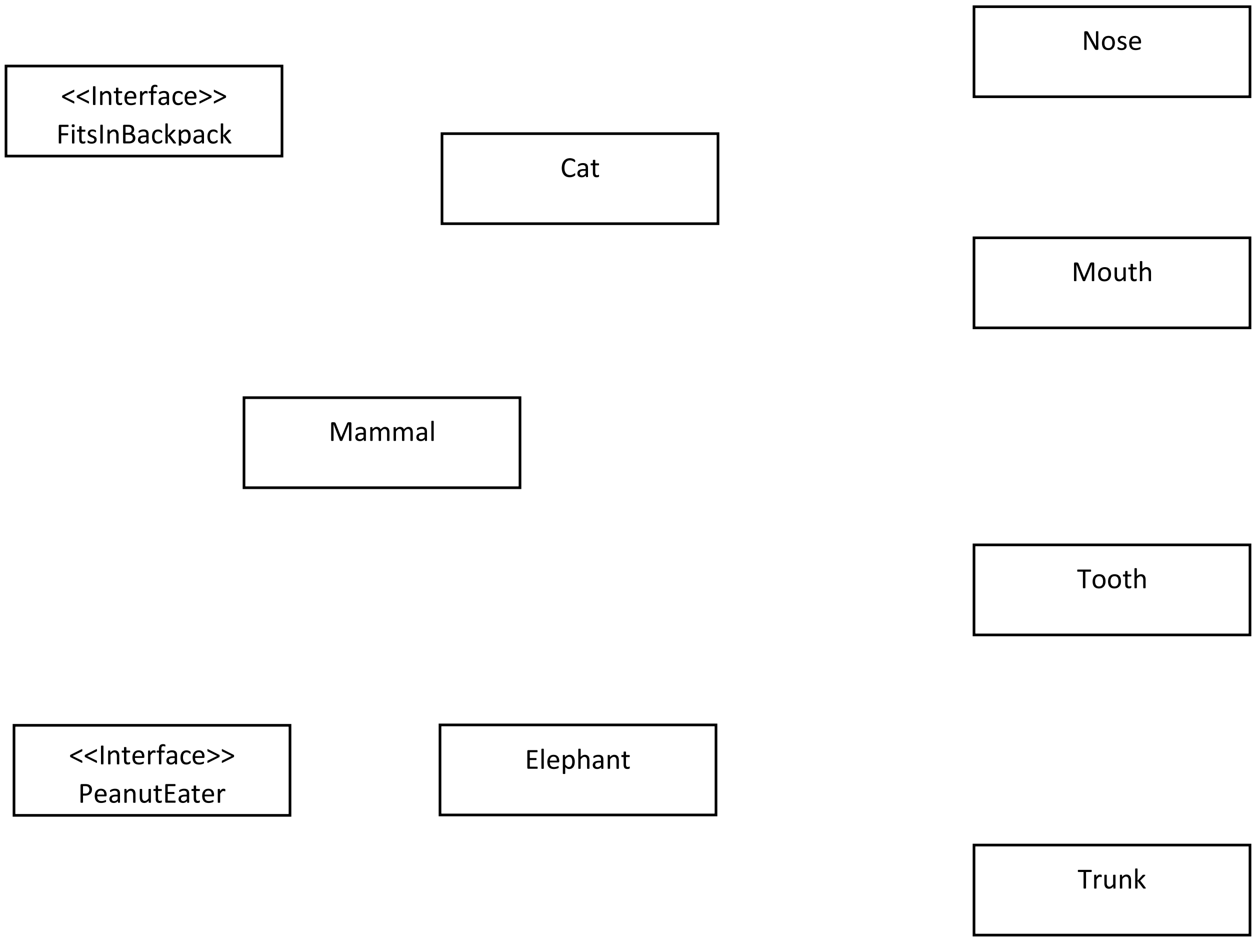
[10+2] Q1) UML

ANSWER ON THIS PAGE

UML notation consists of symbols, line styles, and arrow tips. Use the appropriate symbols for inheritance/generalization, interface implementation, association, aggregation, composition, and dependency to complete the following UML diagrams.

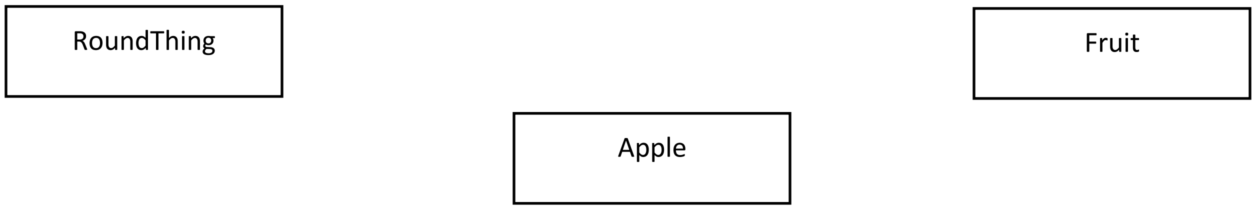
[10] a)

Note: For this Q, an Elephant's nose is its trunk, and cats do not eat peanuts but elephants do.



[0+2] b) Bonus

Note: All apples are fruits. All apples are round things. Not all round things are Fruit, and not all fruit are round things. None of these classes are interfaces.



[16] Q2) MC

ANSWER IN THE ANSWER BOOKLET

1) The *CRC Card* method, used to identify potential objects during design time:
   A) Is a type of UML diagram.
   B) Records class names with *reasons* and *collaborators* to provide easy <u>access</u> to the class.
   C) Records class names *rapidly* and *carefully* on index cards.
   D) Records class names with *responsibilities* and *culprits* to blame when things don't work.
   E) Identifies what classes should do, and identifies what classes may help to do this work.

2) Selecting a method among several methods with the same signature based on the actual object instance (not on the type of the variable containing the object reference) is called:
   A) Dynamic Method Lookup
   B) Encapsulation
   C) Cohesion
   D) Coupling
   E) Inheritance

3) Which statement is <u>**incorrect**</u>?
   A) You can have an *object reference* whose type is an abstract class.
   B) An abstract method of an abstract class has no implementation.
   C) You cannot construct an *object* of an abstract class.
   D) An abstract class cannot have a constructor.
   E) If a class includes abstract methods, then the class itself must be declared as `abstract`.

4) Which statement is <u>**incorrect**</u>?:
   A) In Java, you can throw anything that is-a Object.
   B) In Java, you can throw anything that is-a Throwable.
   C) In Java, you can throw anything that is-a Exception.
   D) In Java, you can throw anything that is-a Error.

5) Which of the following most likely indicates that you have chosen a good name for your class?
   A) The name consists of a single word.
   B) You can tell by the name what an object of the class is supposed to represent.
   C) You can easily determine by the name how many concepts the class represents.
   D) The name is the task the class will perform.

6) Which of the following is a good indicator that a class is overreaching and trying to accomplish too much?
   A) The class has more constants than methods
   B) The public interface refers to multiple concepts
   C) The public interface exposes private features
   D) The class is both cohesive and dependent.

7) Which of the following statements about inheritance is correct?
   A) You can always use a superclass object in place of a subclass object.
   B) You can always use a subclass object in place of a superclass object.
   C) A superclass inherits data and behavior from a subclass.
   D) A superclass inherits only behavior from a subclass.

8) Which of the following statements about a Java interface is **incorrect**?
   A) A Java interface must contain more than one method.
   B) A Java interface defines a set of methods that are required.
   C) A Java interface specifies behavior that a class will implement.
   D) All methods in a Java interface must be public.
9) Which of the following statements about an interface is true?
   A) An interface has both public and private methods.
   B) An interface has methods and instance variables.
   C) An interface has methods but no instance variables.
   D) An interface has neither methods nor instance variables.
10) Which of the following statements is correct about inheritance and interfaces?
   A) A class can extend multiple classes and can implement multiple interfaces.
   B) A class can extend at most one class and can implement at most one interface.
   C) A class can extend multiple classes and can implement at most one interface.
   D) A class can extend at most one class and can implement multiple interfaces.
11) xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   A) xxxxxxxxxxxxxxxxxxxx
   B) xxxxxxxxxxxxxxxxxxxx
   C) xxxxxxxxxxxxxxxxxxxx
   D) xxxxxxxxxxxxxxxxxxxx
12) Which of the following statements about checked and unchecked exceptions is true?
   A) Checked exceptions are handled by the Java runtime.
   B) The compiler ensures that the program is handling unchecked exceptions.
   C) The compiler ensures that the program is handling checked exceptions.
   D) All exceptions that are descendants of RunTimeException are checked exceptions.
13) _____ is often described as the “is-a” relationship.
   A) Inheritance
   B) Aggregation
   C) Polymorphism
   D) Dependency
14) _____ is often described as the “has-a” relationship.
   A) Inheritance
   B) Aggregation
   C) Polymorphism
   D) Dependency
15) When designing classes, if you find classes with common behavior you should ____.
   A) Combine them all into a single class.
   B) Place some common behavior into a superclass.
   C) Place the common behavior into a subclass.
   D) Give them similar names.
16) Which of the following statements about achieving complex layouts in a GUI is **incorrect**?
   A) By giving each panel an appropriate layout manager, the panels can be nested
   B) There are more complex layout managers such as the grid bag layout and group layout
   C) You can use as many panels as you need to organize your components
   D) The order of the nested panel does not matter

[30+3] Q3) Abstract Classes & Inheritance

ANSWER IN THE ANSWER BOOKLET

You are provided with the API of the abstract class ThreeDBody. Do not modify it.

Write a complete class Sphere in the answer booklet. You do not need to label the various parts as "Step 1" to "Step 7."

[1] Step 1) Declare the class Sphere which is a concrete subclass of the abstract class ThreeDBody.

[1] Step 2) Define a constant MAX_RADIUS with a value of 100. MAX_RADIUS should be static, unchangeable, of type int and accessible outside the class Sphere.

[12] Step 3) Implement a constructor that takes exactly three arguments: the colour (of type java.awt.Color) of the sphere, a boolean indicating if the sphere is hollow or not (hollow if true, solid otherwise), and the sphere's radius. If radius is not in the range $1 \leq r \leq MAX\_RADIUS$, then the constructor should throw the unchecked exception java.lang.IllegalArgumentException.

[6] Step 4) Override ThreeDBody's .calculateSurfaceArea abstract method to calculate and return the surface area of the sphere using the formula $SA = 4\,\pi\,r^2$. If the calculated surface area is not at least MIN_AREA (defined in ThreeDBody), then the method throws the checked exception java.lang.Exception.

[6] Step 5) Override the method .equals defined on Object. It should return true if and only if this sphere is equal to another object.

[4+1] Step 6) Override the method .toString defined on ThreeDBody to give a textual representation of the Sphere object. This method returns a string with the name of the class, the colour of the sphere, its radius, and whether the sphere is hollow or solid.

No input or output occurs in the class Sphere.

You may provide private helper methods if you need to, but I don't think that you do.

[0+2] BONUS: given the above, what other method should we override on Sphere? Name it, and provide its signature.

ANSWER IN THE ANSWER BOOKLET

[10] Q4) Tracing – Inheritance and Polymorphism

ANSWER IN THE ANSWER BOOKLET

Given:

```
public class Hamburger {
   private final String typeOfBread;

   public Hamburger(){
      this("hamburger bun");
   }
   public Hamburger(String typeOfBread){
      this.typeOfBread = typeOfBread;
   }
   public String toString() {
      return "\n" + typeOfBread + " bread";
   }
}

public class FishBurger extends Hamburger {
   private final String[] ingredients = {"pollock", "mayo"};

   public FishBurger (String str) {
      super(str);
   }

   public String toString() {
      String s = super.toString();
      s += " with ";
      for (String ing: ingredients)
         s += ing + " ";
      return s;
   }
}

public class CheeseBurger extends Hamburger {
   private final String cheese;

   public CheeseBurger() {
      super("white");
      this.cheese = "American cheese";
   }
   public CheeseBurger(String cheese) {
      super();
      this.cheese = cheese;
   }
}
```

```
public class PapaCheeseBurger extends CheeseBurger {
   public PapaCheeseBurger() {
      super();
   }

   public PapaCheeseBurger(String str) {
      super(str);
   }
   public String toString() {
      return super.toString() + " papa big and cheesy";
   }
}

public class Menu {
   public static void main(String[] args) {
      Hamburger a, b, c;
      CheeseBurger d;
      a = new FishBurger("white");
      b = new CheeseBurger("cheddar");
      c = new CheeseBurger();
      d = new PapaCheeseBurger();

     System.out.println(a.toString());
     System.out.println(b.toString());
     System.out.println(c.toString());
     System.out.println(d.toString());
  }
}
```

ANSWER IN THE ANSWER BOOKLET

[3] a)

Draw the UML diagram depicting the relationship between the four classes of burgers (exclude Menu).

[7] b)

Give the output of executing the main method of the class Menu.

ANSWER IN THE ANSWER BOOKLET

[22+1] Q5) Interfaces

Given:

```
public interface Closable {
     public void close();
}

public interface Openable extends Closable {
     public void open();
}

public class Door {
     public enum State {OPEN, CLOSED};
     private State state = State.CLOSED;
     /* … some stuff */
     public void setState(State aState) {
          state = aState;
     }
}
```

What changes would need to be made to the class door to allow this statement to be legal (to compile):

```
Openable op = new Door();
```

ANSWER IN THE ANSWER BOOKLET

[7+1] a)

Provide a new implementation of Door that will make this assignment legal.

[5] b)

Provide a full UML diagram for these classes/interfaces, including the methods and variables shown.

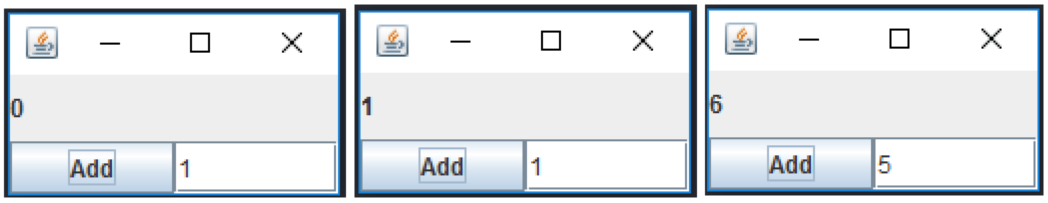
[10] c)

After making the above legal, for each of the following, indicate whether each line would be legal or illegal. Each line is independent. (-1 for incorrect answers, minimum mark of 0):

```
i)    Closeable cl = new Door();
ii)   Object ob = new Door();
iii)  Openable op = new Door() ; op.close();
iv)   Closable cl = new Door() ; cl.open();
v)    Object ob = new Door() ; ob.open();
vi)   Closable cl = new Door() ; ((Openable)cl).close();
vii)  Closable cl = new Door() ; ((Closable)cl).open();
viii) Closable cl = new Door() ; Openable op = cl;
ix)   Openable op = new Door() ; Closable cl = op;
x)    Object ob = new Door(); Closable cl = (Openable) ob;
```

[20+1] Q6) GUIs

Consider these 3 screenshots of a GUI program written in Java 8:



This window opens with the value “0” in the label, and the number “1” in the text field. Activating the “Add” button (through a mouse click or otherwise) will add the value in the text field to the value in the label (ie: 0+1=1). The user may change the value in the text field to a different integer. After doing so, activating the “add” button will then add that value to the label (ie: 1+5=6); **except** for when the user enters “0” into the text field, then activating the “add” button will reset the value in the label to “0”. Changing the value in the text field on its own does nothing.

ANSWER IN THE ANSWER BOOKLET

Reproduce this program by following these steps:

[Note: you do not have to do any error handling. Assume that the user only enters valid integers.]

Step 0) Assume that your class alread includes the following main method (you do not have to write it):

```
public static void main(String[] args) {
  JFrame f = new GUI();
  f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  f.pack();
  f.setVisible(true);
}
```

[1] Step 1) Write the declaration for your class. Name it “GUI” and make it a JFrame. For the sake of brevity, all logic will be contained in this one class.

[2] Step 2) Declare any instance variables that you may need. This question can be solved with no instance variables, but I would consider it bad style to do it that way. You should not need more that 4 instance variables. Don’t try to be clever – do things in the most straightforward way.

[4] Step 3) Initialize the GUI controlls (the input an outputs) by instantiating/creating them in memory. Set their default display values, and initialize any other instance variable(s) that are required.

[6] Step 4) Do the layout for the program by setting any necessary layout manager(s), adding your controlls, and any other necessary component(s).

[7+1] Step 5) Create and attach any necessay listener(s). +1 bonus for using a lambda expression.

That’s it for the GUI!

[12] Q7) Tracing – Exceptions

Given:

```
public class Etrace {

  private static void a(int x) {
    try {
      System.out.println("1");
      b(x);
      System.out.println("2");
    } catch(RuntimeException e) {
      System.out.println("3");
    } finally {
      System.out.println("4");
    }
    System.out.println("5");
  }

  private static void b(int x) {
    try {
      System.out.println("6");
      if(x < 0) {
        throw new IllegalArgumentException();
      } else if(x==0) {
        throw new RuntimeException();
      }
      System.out.println("7");
    } catch (IllegalArgumentException e) {
      System.out.println("8");
    } finally {
      System.out.println("9");
    }
    System.out.println("0");
  }

  public static void run(int x) {
    System.out.println("* " + x + " *");
    a(x);
  }
}
```

ANSWER IN THE ANSWER BOOKLET

What is the output from calling:

a) Etrace.run(-1)
b) Etrace.run(0)
c) Etrace.run(1)

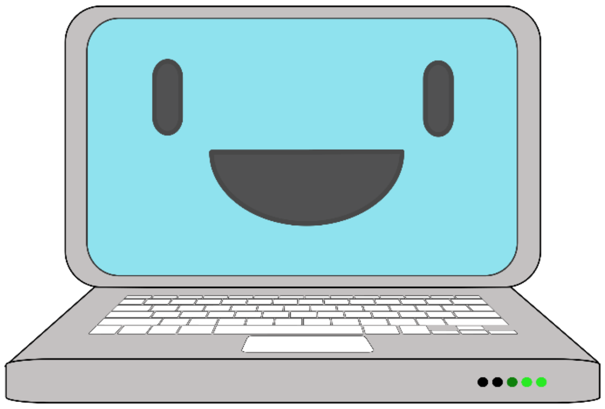
[3] Q8) BONUS – Threads

Consider:

```
public class BuggyThread {
  // NOTE: contains some errors
  // NOTE: may be compile or logic errors

  private final static int NUM_THREADS = 10;

  public static void main(String[] args) {

    Integer out = new Integer(0);

    Thread ts[] = new Thread[NUM_THREADS];
    for(int i = 0; i < NUM_THREADS; i++) {
      ts[i] = new Thread(new ToRunRunnable(out));
      ts[i].start();
      ts[i].join();
    }
    assert out.equals(NUM_THREADS);
  }
}

// NOTE: ToRunRunnable is-a Runnable. It increments the passed
Integer object once in a thread-safe way
```

Identify the errors in the above given code.

ANSWER IN THE ANSWER BOOKLET



Enjoy your weekend!