Computer Science 51G
Fall 2014

Midterm Examination
13 October, 2014

Question	Points	Score	Description
1	10		Program analysis
2	15		Class design
3	15		GUIs and ActiveObjects
4	10		Short Answers
Total	50		

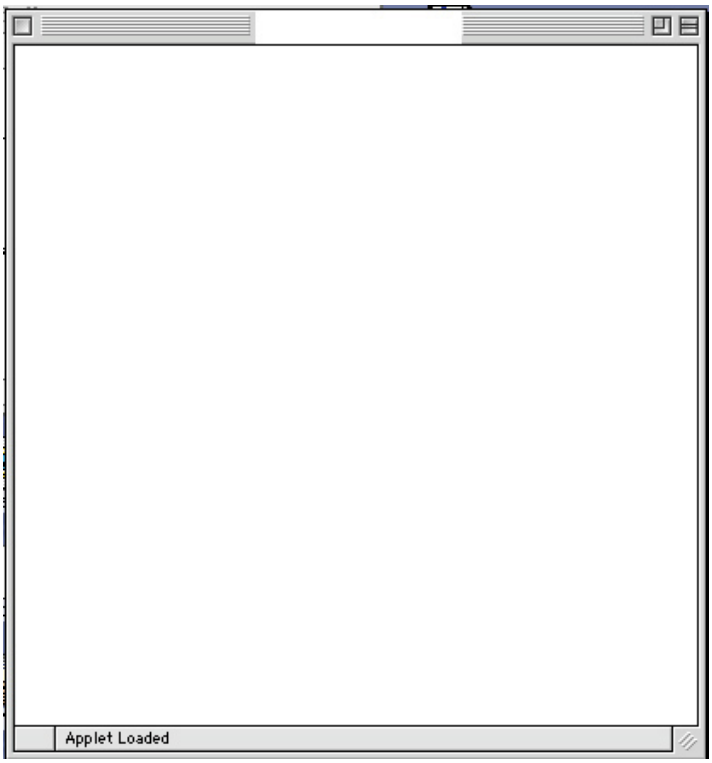
This examination is closed book. You have 55 minutes to complete the exam.
You do not need to include comments in your code unless you feel it is necessary for us to understand your solution.

Your Name (Please print) _____

1. **Program analysis**:

The following is a simple but complete Grace program that does nothing more than draw a picture on the canvas. Draw a sketch of the picture that will be produced on the canvas below.

```
dialect "objectdraw"

def mystery: GraphicApplication = object {
  inherits graphicApplication.size(100, 100)

  var x: Number
  var y: Number := 0
  while {y < 100} do {
    x := 0
    while {x <= y} do {
      framedOval.at(x @ y) size (25, 25) on (canvas)
      x := x + 25
    }
    y := y + 25
  }

  startGraphics
}
```



2. **Class design**:

It is about that time of year when Gus the Ghost returns to his favorite haunt – the CS 51G lab. We want you to write a `Ghost` class for Gus. Here is his picture:

Gus's body can be drawn as a gray circle and overlaid gray rectangle. The rectangle's height is twice its width, and it should cover half of the circle. We provide constants for some of these dimensions (and the ghost's color) for you in the starter code below. Gus has two eyes- the tops of the eyes are at the same height as the top of the rectangle, and the distance between each and the edges of the ghost is 1/4 the rectangle's width. We want you to implement a class with several methods for `Ghost`.

First, write a `ghost` class that takes two parameters:

- A point representing the upper left corner of the ghost, and
- a canvas.

Since Gus can move around the lab, you should also write a `moveBy` method that moves a Ghost by an amount specified by the caller of the method, i.e. moveBy should have two parameters, a `dx` and a `dy`. In addition, we want you to write a `contains` method to test if a point is inside the `Ghost`.

Gus turns invisible to sneak up on unsuspecting students. He does this by making his body disappear and reappear. However, you can always see his eyes. The method to do this in the `Ghost` class is `visibleBody:=().` If `myGhost` has type `Ghost`, then after evaluating `myGhost.visibleBody:=false`, `myGhost` should appear like this in the canvas:

Requesting `myGhost.visibleBody:=true` will make the ghost body reappear, and the ghost will look like it did in the first picture above. Fill in the missing details from the `ghost` class below. We have created the eyes for you, but not the body or head of the ghost.

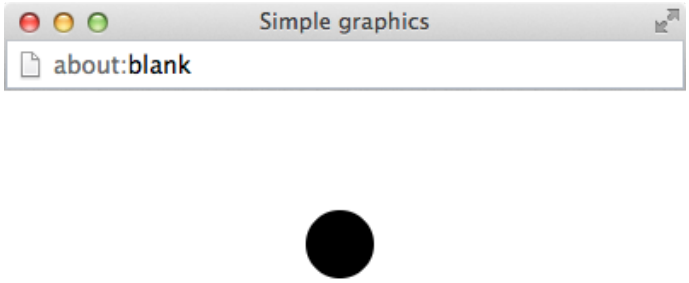
```
type Ghost = {
  moveBy(dx: Number, dy: Number) -> Done
  contains(pt: Point) -> Boolean
  visibleBody:=(tval: Boolean) -> Done
}
class ghost. at(pt: Point) on (canvas: DrawingCanvas)
```

```
                              -> Ghost
def bodyWidth: Number = 40
def bodyHeight: Number = bodyWidth * 2
def ghostColor: Color = lightGray
def eyeSize: Number = 8
def eyeInset: Number = 10

def leftEye: Graphic2D =
  filledOval.at(pt+(eyeInset@eyeInset))
             size (eyeSize,eyeSize) on (canvas)
def rightEye: Graphic2D =
  filledOval.at
            (pt+((bodyWidth-eyeInset-eyeSize)@eyeInset))
            size(eyeSize,eyeSize) on (canvas)









method moveBy(dx: Number, dy: Number) -> Done {




}
```

```
  method contains(pt: Point) -> Boolean {




  }
  method visibleBody:=(tval: Boolean) -> Done {









  }
}
```

3. **GUI & Animation Programming**:

For this problem, you are to program a simple game that uses a button and an animated object. When the user clicks the mouse, a ball appears at the location of the click and starts falling down the screen. There is a button at the bottom of the window labeled "Push to punch ball up". Each time the user pushes the button the ball is pushed up the screen by 100 pixels. The game continues until the ball falls completely off the canvas (i.e., the top of the ball is no longer visible). In the full version of the game, we would keep track of how close to the bottom of the screen we get, but we'll ignore that here.



The skeletons of the main program and class needed to build this game are provided below. Your task is to fill in the missing code for each. For the class fallingBall you need only to insert code for the method punchIt and start. For the main program, saveMe, you need to insert the code to create, install, and associate an action with the button.

```
dialect "rtobjectdraw"
import "animation" as animator

type AnimatedPuncher = {
  start -> Done
  punchIt -> Done
}

// used to initialize theBall to prevent crash
def emptyPuncher: AnimatedPuncher = object {
  method start -> Done {}
  method punchIt -> Done {}
}

// class to create a falling ball that can be “punched” up
class fallingBall.at(pt: Point) radius (size: Number)
            on (canvas: DrawingCanvas) -> AnimatedPuncher {
  def punchDistance: Number = 100
  def pauseTime: Number = 30
  def dropDistance: Number = 2
  def ball: Graphic2D =
          filledOval.at(pt) size (2*size, 2*size) on (canvas)

  // add code to “punch up” ball
  method punchIt -> Done {









  }

  // start the animation
  method start -> Done {









  }
}
```

```
def saveMe: GraphicApplication = object {
  inherits graphicApplication.size(400,400)
  var theBall: AnimatedPuncher := emptyPuncher

// Insert code to create and use the button









  method onMousePress(pt: point) -> Done {
    theBall := fallingBall.at(pt) radius (20) on (canvas)
    theBall.start
  }

  startGraphics
}
```

4. Short answers:

a. Suppose a Grace program includes the following code:

```
class cText.new  {
  def x = 7
}

def c = cText.new
print (c.x)
```

When this is executed, the system prints the following error message:

```
NoSuchMethod at line 6 of test: Requested confidential method
'x' on object an object from outside.
```

Please explain what the problem is, and provide the simplest fix to the code in class cText that would allow the print statement to execute without error.

b. Suppose that in our Frogger program that each time a vehicle `aCar` went off the screen the program executed:

```
aCar.visible := false
```

rather than

```
aCar.removeFromCanvas
```

What is the difference between these two statements and what practical consequences would this change have?

c. The following code fragment is correct, but not as elegant or as efficient as it should be. Please improve the code so that it does what the original does, but is more elegant or efficient. Write the corrected code below the original code. You may assume that `inABox` is declared elsewhere as a variable of type Boolean.

```
method onMousePress(point: Point ) -> Done {
  if ( box1.contains(point) ) then {
     inABox := true;
  } elseif ( box2.contains(point) ) then {
     inABox := true;
  } else if ( box3.contains(point) ) then {
      inABox := true;
  } else {
      inABox := false;
  }
}
```

d. In games like that described in problem 3 of this exam, we often want to keep track of the score so far to motivate the player to continue playing. Suppose that we have declared a variable `highScore` to use in the game. Please write a method to update the `highScore` variable given that the latest score recorded is provided in a parameter `newScore`. Please write the body of the method that will ensure that after execution, the value of `highScore` is the highest score seen so far. You may not assume that you have a predefined method that computes the maximum of two values.

```
// After this method is executed, the value of
// highScore is the largest of its previous value
// and the value of newScore
method updateHighScore(newScore: Number) -> Done {




}
```