# 10 Revision

## Practical Test Practice

(a) Revision for text functions (such as text(), textAlign(), textSize(), etc) and the line() function:

Write a program that draws a ruler. The ruler should be 10 cms long (you can make 1cm equal to 100 pixels on screen) with major tick marks at 0, 5 and 10 cms, and minor tick marks every cm. Every tick mark should be labeled.

(b) Use the textWidth() function to write a program that draws a string on the screen with a correctly sized box around the string.

(c) Write a class to draw a grid that fills the sketch. The number of cells across and down should be set by properties of a grid class.

(d) Look up the nf() function in the Processing reference. Write a program that prints PI, PI/2 and PI/3 to one, two, four and eight decimal places.

(e) Revise 2D arrays: modify your grid program from (b) so that every cell has a different colour. The colours should be stored in a 2D array, i.e. you need a field like this:

private color[][] cellColours;

(f) Write a program that prints out a noughts and crosses board. Each position on the board should have a cross, a nought, or be empty. Use a 2D array to store the board contents.

(g) Write a program to draw a map. A map is a 10x10 grid. Each cell of the grid can be occupied by a sea, land, mountain or beach tile. Create an abstract Tile class that is the superclass of concrete classes Sea, Land, Mountain and Beach. The four classes differ only in how they are drawn. Create a Map class that stores the Tiles of the map in a 2D array, and draws them out. Draw the program as a UML class diagram before you start.

## Final Exam

Q1. Study this class:

class Histogram {

public char symbol = '\*';

protected String[] categories;

protected int[] frequencies;

public Histogram(int numCategories) {

categories = new String[ numCategories ];

frequencies = new int[ numCategories ];

for (int index=0; index<numCategories; index++) {

categories[ index ] = "unlabeled";

frequencies[ index ] = 0;

}

}

public void setCategory(int index, String name, int frequency) {

categories[ index ] = name;

frequencies[ index ] = frequency;

}

public String toString() {

String result = "";

for (int index=0; index<categories.length; index++) {

result+=categories[index]+": ";

result+=repeatSymbol(frequencies[index]);

result+="\n";

}

return result;

}

protected String repeatSymbol(int numTimes) {

String result = "";

for (int index=0; index<numTimes; index++)

result += symbol;

return result;

}

}

(a) Identify (i) the constructor, (ii) the fields and (iii) the methods.

(b) Predict the output of the following code fragments, or state that there will be an error. If there is an error, state the line on which the error occurs and explain.

Histogram hist1 = new Histogram(3);

hist1.setCategory(0, " Apples", 4);

hist1.setCategory(1, "Bananas", 2);

hist1.setCategory(2, "Oranges", 5);

println( hist1 );

Outcome:

Histogram hist2 = new Histogram(4);

hist2.setCategory(0, "Chocolate", 10);

hist2.symbol='!';

hist2.setCategory(1, " Onions", 3);

println( hist2 );

Outcome:

Histogram hist3 = new Histogram();

hist3.setCategory(0, "Cars ", 3);

hist3.setCategory(1, "Bikes", 5);

println( hist3 );

Outcome:

Histogram hist4 = new Histogram(2);

hist4.setCategory(0, "Poppies", 10);

hist4.setCategory(2, "Roses", 3);

println( hist4 );

Outcome:

Histogram hist5 = new Histogram(4), hist6;

hist5.setCategory(0, "Chocolate", 10);

hist6=hist5;

hist6.symbol='!';

hist6.setCategory(1, " Onions", 3);

println( hist6 );

Outcome:

Q2. A cumulative histogram is a histogram in which each value is added to the sum of the values that have gone before. For example if there are 3 Apples, 2 Bananas and 6 Oranges, then the numbers in the normal histogram will be (3,2,6) but in a cumulative histogram they will be (3,5,11).

Create a subclass of Histogram called CumulativeHistogram. You **only** need to override the toString() method to solve this problem.

Here is a template for the class:

class CumulativeHistogram extends Histogram {

public CumulativeHistogram(int numCategories){

super(numCategories);

}

public String toString() {

// \*\*\*\*\*\*\*\*\* Fill in this method

}

}

And here is some test code:

Histogram hist = new Histogram(3);

hist.setCategory(0, " Apples", 4);

hist.setCategory(1, "Bananas", 2);

hist.setCategory(2, "Oranges", 5);

println( hist );

CumulativeHistogram hist2 = new CumulativeHistogram(3);

hist2.setCategory(0, " Apples", 4);

hist2.setCategory(1, "Bananas", 2);

hist2.setCategory(2, "Oranges", 5);

println( hist2 );

And its output:

Apples: \*\*\*\*

Bananas: \*\*

Oranges: \*\*\*\*\*

Apples: \*\*\*\*

Bananas: \*\*\*\*\*\*

Oranges: \*\*\*\*\*\*\*\*\*\*\*

Q3. Design the classes, properties, methods, and class relationships for the following problems. Present your answers as a UML class diagram.

(a) *You are to write a program to make slide shows, similar to but simpler than Powerpoint. Each slide should have a variety of components on it, such as text boxes and images. Slide shows should also have options to create themes, for example common transitions between slides and common backgrounds and styles. Add functionality to show a slideshow and navigate back and forth between slides.*

(b) *Write a system for storing information for a bus company. Bus companies need to store information about buses (such as registration number, age, size, manufacturer, hours used etc) and also about drivers (name, address, years of service etc). The bus company also has different routes (e.g. number 18 – Te Rapa, number 04 – Flagstaff), and a bus with a driver assigned must run along each route every half hour.*

Q4. Study this inheritance hierarchy:

abstract class Vehicle {

abstract public void drive();

}

class Car extends Vehicle {

public String make, model;

public void drive() {

println("a car drives");

}

}

class Minivan extends Car{

public float numSeats;

public void drive() {

println("a minivan drives");

}

}

class SportsCar extends Car{

public float topSpeed;

public void drive() {

println("a sportscar drives");

}

}

abstract class Bicycle extends Vehicle {

public String manufacturer;

public void drive() {

println("a bike is ridden");

}

}

class MountainBike extends Bicycle{

}

class Tandem extends Bicycle{

}

(a) Draw as well as you can the corresponding UML diagram

(b) Predict the output of the following code fragments, or state that there will be an error. If there is an error, state the line on which the error occurs and explain.

SportsCar myCar = new SportsCar();

myCar.drive();

myCar.make = "Ferrari";

Outcome:

Bicycle bike = new Bicycle();

bike.drive();

bike.manufacturer = "24Seven";

Outcome:

Tandem tandem = new Tandem();

tandem.manufacturer = "24Seven";

tandem.drive();

Outcome:

Vehicle vehicle = new MountainBike();

vehicle.drive();

Outcome:

SportsCar sporty = new SportsCar();

sporty.topSpeed = 210;

println( "sporty's top speed is "+sporty.topSpeed );

Outcome:

SportsCar sporty2 = new SportsCar();

Car car = sporty2;

car.topSpeed = 180;

println( "sporty2's top speed is "+car.topSpeed );

Outcome:

SportsCar sporty3 = new SportsCar();

Car car = sporty3;

((SportsCar)car).topSpeed = 250;

println( "sporty3's top speed is "+((SportsCar)car).topSpeed );

Outcome:

Minivan van = new Minivan();

van.numSeats = 8;

van.make = "Mazda";

van.model = "Estima";

van.drive();

Outcome:

Minivan van2 = new Minivan(), van3;

van3 = van2;

van3 = null;

van2.drive();

van3.drive();

Outcome:

Minivan van4 = new Minivan();

Car someCar = van4;

someCar.make="Toyota";

someCar.drive();

println("van4's make = " + van4.make );

van4.drive();

Outcome:

Q5. Imagine that you have written a function called randomPattern() that draws a certain number of random circles and a certain number of random squares on the screen. The header of the function is:

void randomPattern(int numCircles, int numSquares);

If there is something wrong with one of the parameters (i.e. if its negative), then the function should write out an error message to the console and do nothing.

Since there is effectively an infinite number of different *test cases* (a test case is one call to the function such as randomPattern(4,5)), how would go about testing this function to ensure that it is correct? That is, what would your best set of test cases be?

Q6. In the PVector class, the method get() returns a *copy* of the PVector. Explain the difference between:

PVector a = new PVector(10,12);

PVector b = a;

and

PVector a = new PVector(10,12);

PVector b = a.get();

Q7. Consider this class definition, which adds a new field q to the PVector class:

public class MyVector extends PVector {

public float q;

public MyVector(float x, float y, float q) {

super(x, y);

this.q=q;

}

}

(a) What does the call to super() do? Why is there no field for x and y in the class?

(b) What is the difference between this.q and q?

Suppose you (try to) run the following code:

MyVector mine = new MyVector(230,120,33);

PVector yours = new PVector(100,100);

println(mine.dist(yours));

println(yours.dist(mine));

(c) Will it work?

(d) If there is no method dist() defined for class MyVector, what does it mean to call mine.dist()?

(e) The call to mine.dist(yours) expects a PVector as a parameter to the dist() method, but yours is not a PVector. Why does this work/not work?