# COMP258 Exam Questions

1. Consider the following program that loads, adjusts and saves an image:

void setup() {

PImage image = loadImage("dog.jpg"); // Line 1

println(image.width+" x "+image.height); // Line 2

image.resize(100,100); // Line 3

println("format is "+image.format); // Line 4

image.save("dog\_resized.jpg"); // Line 5

}

Clearly specific the line numbers in which (i) fields/properties appear, and (ii) methods are called.

A: fields/properties are accessed on lines 2 & 4; methods are called on lines 3 & 5.

2. Consider the following program intended to print out an ellipse at position (100,200) on the sketch:

PVector position; // Line 1

void setup() {

size(300,300);

PVector position = new PVector(100, 200); // Line 2

println(position.x); // Line 3

println(position.y); // Line 4

}

void draw() {

fill(255);

ellipse(position.x, position.y, 10, 10); // Line 5

}

There is an error in this code. You are to (i) explain the error by stating the cause of the error and the line on which it occurred, and (ii) correct the error by making the *minimal* change needed to the above example that makes the program draw the ellipse correctly.

A: (i) there are two position vectors created, one outside setup (global scope, line 1) and one inside setup (local scope, line 2); the global position is null; a null pointer reference error occurs therefore on line 5; (ii) deleting the first PVector from line 2 will correct the program.

3. Consider the following class along with sample code showing how to use the class:

class Die {

private int numFaces; // Line 1

public Die(int numFaces){ // Line 2

this.numFaces = numFaces; // Line 3

}

public int roll() { // Line 4

return (int)random(numFaces) + 1; // Line 5

}

}

void setup(){

Die d6 = new Die(6); // Line 6

println(d6.roll()); // Line 7

}

(i) Which lines contain the definition of a method? On which line is the method called?

(ii) Which lines contain the definition of a constructor? On which line is the constructor called?

(iii) Suppose a third line is added to the setup() method after line 7, specifically:  
d6.numFaces = 8;

Will the program compile or not compile? If the program compiles, will it run correctly or will there be an error? If there will be any kind of error, explain.

A: (i) method defined on 4-5, called on 7; (ii) constructor defined on 2-3, called on 6; (iii) the program will not compile because there is no public property numFaces

4. Study the following class representing a clock very carefully:

class Clock {

public int hour=12, minute=0;

public void addMinute() {

minute++;

if (minute>59) {

minute=0;

hour++;

if (hour>12)

hour=1;

}

}

public void addNMinutes(int n) {

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

addMinute();

}

public String toString(){

String result=hour+":";

if (minute<10) result+="0";

result+=minute;

return result;

}

}

Given your understanding of the class, predict the output of this fragment of code:

void setup() {

Clock first = new Clock(),second=new Clock();

first.hour=3;

first.minute=45;

second.hour=12;

second.minute=first.minute+10;

println(first.toString());

println(second.toString());

first.addNMinutes(20);

second.addNMinutes(first.minute);

println(first.toString());

println(second.toString());

}

A:

3:45

12:55

4:05

1:00

5. Consider the following brief:

*A library system needs to keep track of books and library patrons. Books have a title, author, publisher and publication date. Library patrons have a name, address and patron ID number. All library employees are automatically patrons, and have a library branch that they work at. Patrons can check out books from the library (up to a maximum of ten books), and the system needs to keep track of which patron has checked out which book. Every time a book is issued, an issuing and a due date should be stored.*

Draw a class diagram reflecting this situation as accurately as possible. Your diagram should have

* class names
* property names
* association and inheritance relationships
* multiplicities

Types for the properties, and methods for the classes, are *not* required.

A:



6. Study the following program carefully:

Board board;

void setup() {

board=new Board();

board.addPiece('X',0,0);

board.addPiece('O',0,2);

board.addPiece('X',1,1);

board.addPiece('O',1,2);

board.addPiece('O',2,1);

board.addPiece('X',2,2);

}

void draw() {

board.draw();

}

class Piece {

private char symbol;

private PVector position;

**public Piece(char symbol, float x, float y) {**

**// Missing code**

**}**

public void draw() {

textAlign(CENTER, CENTER);

fill(255);

textSize(80);

text(symbol, position.x, position.y);

}

}

class Board {

private Piece[][] pieces;

public Board() {

pieces = new Piece[3][3];

}

public void addPiece(char symbol, int x, int y) {

pieces[x][y] = new Piece(symbol, 50+x\*100, 40+y\*100);

}

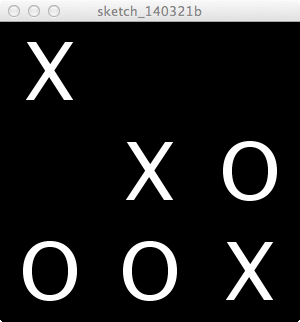
**public void draw() {**

**// Missing code**

**}**

}

The output of this program is the following sketch:



There is missing code in two places in the above example, specifically:

* the constructor for the Piece class
* the draw method for the Board class

Write down completed versions of both of these items, assuming that pieces may be added in any configuration inside setup().

A:

public Piece(char symbol, float x, float y) {

this.symbol=symbol;

position = new PVector(x, y);

}

public void draw() {

for (int x=0; x<3; x++)

for (int y=0; y<3; y++)

if (pieces[x][y]!=null)

pieces[x][y].draw();

}

7. Consider the following class:

class CircleGroup {

public PVector[] positions;

public float[] sizes;

public void draw() {

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

ellipse(positions[index].x, positions[index].y,

sizes[index], sizes[index]);

}

}

A *normal* test case for this class is the following code:

CircleGroup group = new CircleGroup();

group.positions = new PVector[3];

group.positions[0] = new PVector( 100, 100 );

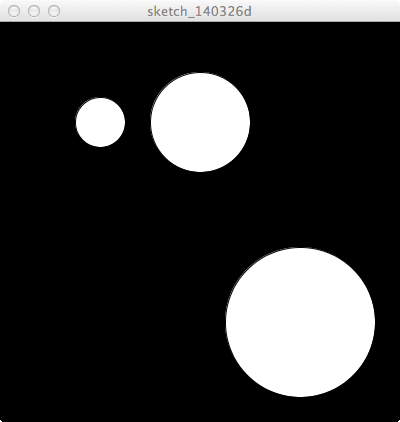
group.positions[1] = new PVector( 200, 100 );

group.positions[2] = new PVector( 300, 300 );

group.sizes = new float[] {50, 100, 150};

group.draw();

which happens to produce this output:



Consider now *erroneous* test cases for this class that cause the program to crash/throw an exception at runtime. Write down all the erroneous test cases for this class that you can think of, along with the error they will cause.

A: positions may be null (null pointer exception); sizes may be null (null pointer exception); length of position exceeds length of sizes (array index out of range error)

8.