Escola Superior de Tecnologia e Gestão Instituto Politécnico da Guarda	**ENUNCIADO DE AVALIAÇÃO**	Modelo PED.002.02

Curso	**Engenharia Informática**				*Ano letivo*	2020/2021		
Unidade curricular	**Computação Gráfica**							
Ano curricular	3º	*Semestre*	1º S	*Data*	07/01/2021	*Duração*	2h00	

FREQUÊNCIA

Teste sobre os tópicos do Capítulo 1 ao Capítulo 4. O teste vale 30% da nota final.

1. **(.5)** Descreva a diferença entre as duas subáreas da Computação Gráfica: Síntese de Imagem e Processamento de Imagem.

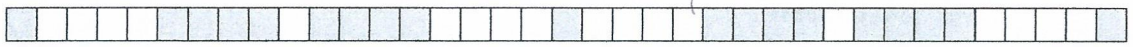
2. **(3.5)** Escreva uma função em Java, com o nome drawArc e com os parâmetros que considerar adequados, para desenhar um arco de circunferência entre um ângulo A e um ângulo B (passados como parâmetros), usando as equações paramétricas da circunferência.

$$x = x_0 + r \times \cos(t)$$
$$y = y_0 + r \times \sin(t)$$
$$A \leq t \leq B$$

3. **(.5)** Considere o seguinte código para configurar o traço de desenho.

```
stroke = new BasicStroke(10, BasicStroke.CAP_BUTT, BasicStroke.JOIN_BEVEL, 0,
dashArray, dashPhase);
```

Indique quais os valores de dashArray e dashPhase para obter o seguinte padrão



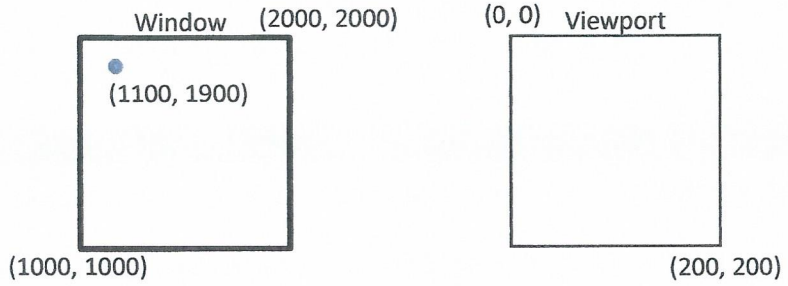
4. **(2)** Desenhe um esboço que represente o resultado do seguinte código. Considere que o painel tem um tamanho de 200x200. Indique no esboço as coordenadas de todos os pontos de controle.

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2 = (Graphics2D) g;
  g2.translate(getWidth()/2, getHeight()/2);
  GeneralPath p = new GeneralPath(GeneralPath.WIND_EVEN_ODD);
  p.moveTo(-50f, -50f);
  p.curveTo(-80, -25, -20, 25, -50, 50);
  p.lineTo(50, 50);
  p.curveTo(20, 25, 80, -25, 50, -50);
  p.quadTo(0, -90, -50, -50);
  p.closePath();
  GradientPaint paint = new GradientPaint(-50, -50, Color.BLACK, -25, -25,
Color.WHITE, true);
  g2.setPaint(paint);
  g2.fill(p);
}
```

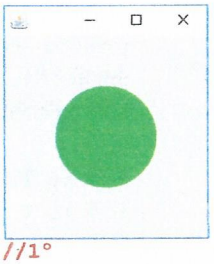
5. **(1.5)** Considere que se pretende aplicar um fator de escala *E* a um objeto 2D qualquer, mas de modo a que as coordenadas do centro do objeto (*Xc*, *Yc*), não se alterem com a transformação. Isto é, pretende-se que o objeto fique centrado no mesmo ponto, após a aplicação da transformação.
Apresente a matriz final 3x3 da transformação composta que ao ser multiplicada pelos pontos do objeto produz o resultado desejado.

$$T = \begin{bmatrix} 1 & 0 & Xc \\ 0 & 1 & Yc \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -Xc \\ 0 & 1 & -Yc \\ 0 & 0 & 1 \end{bmatrix}$$

6. (1) Considere um ponto de coordenadas (1100, 1900) definido na Window da figura. Quais as coordenadas desse ponto na Viewport da mesma figura? Indique todos os cálculos.



7. (3) Complete o código seguinte de modo a implementar uma interação onde o circulo da imagem ao lado alterne a sua cor entre verde e vermelho, sempre que o utilizador clicar com o rato em cima desse mesmo circulo. O código em falta pode consistir em mais de uma instrução. Apresente o resultado como uma lista:



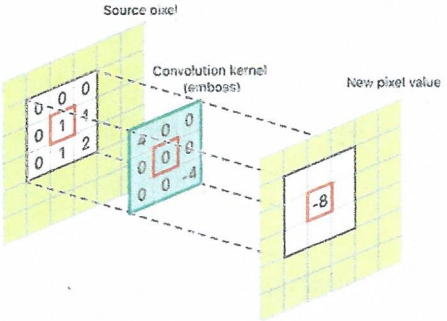
1º -

```
...
class PanelExInteraction extends JPanel implements ... { //1º
  Shape shape = new Ellipse2D.Double(200 - 100, 200 - 100, 200, 200);
                                                          //2º
  public PanelExInteraction() {
    setPreferredSize(new Dimension(400, 400));
                                                          //3º
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
                                //4º More than one instruction
    g2.fill(shape);
  }

  public void mousePressed(MouseEvent e) {
                                //5º More than one instruction
  }
  . . .
}
```

8. (2) A operação de convolução é usada em processamento de Imagem para aplica um dado filtro a uma imagem. O filtro é representado por uma matriz chamada Kernel. Complete o código da seguinte função de modo a que esta aplique a operação de convolução ilustrada na imagem.




```
private                    filter(BufferedImage imgIn) { //1º
  float[] data =                                        //2º
  Kernel ker =                                          //3º
  BufferedImageOp op =                                  //4º
  return                                                //5º
}
```

9. **(2.5)** Desenhe um esboço que represente o resultado do seguinte código. Indique no esboço os eixos do sistema de coordenadas.

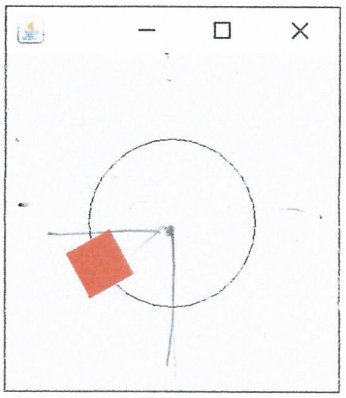
```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2 = (Graphics2D) g;
  int w = this.getWidth();
  int h = this.getHeight();
  int l = 100;
  BufferedImage image = getImage(("test2d/images/Smile.jpg"));
  Paint paint = new TexturePaint(image, new Rectangle2D.Double(-2*l, -l, 2*l,
  2*l));
  Area a = new Area(new Rectangle2D.Double(-l, -l, 2*l, 2*l));
  Area b = new Area(new Rectangle2D.Double(-l/2, -l/2, l, l));

  AffineTransform at = new AffineTransform();
  at.rotate(Math.toRadians(45));
  b=b.createTransformedArea(at);
  a.subtract(b);

  g2.translate(w/2, h/2);
  g2.rotate(Math.toRadians(45));

  g2.setPaint(paint);
  g2.fill(a);
  g2.setColor(Color.black);
  g2.draw(a);
}
```

10. **(3.5)** Complete o seguinte código para criar uma animação onde um quadrado vermelho se move ao longo da órbita circular de cor preta.



```
class MyPanel extends JPanel implements                    {  // 1°
  float ang = 0;

  public MyPanel() {
    setPreferredSize(new Dimension(400, 400));
                                                              // 2°
                                                              // 3°
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;
                                                              // 4°
    g2.drawOval(-100, -100, 200, 200);

    Shape s = new Rectangle2D.Double(-30, -30, 60, 60);
    AffineTransform tx = new AffineTransform();
    tx.                                                       // 5°
    tx.                                                       // 6°
    s =                                                       // 7°

    g2.setPaint(Color.RED);
    g2.fill(s);
  }

  public void          {                                    // 8°
    while (true) {
                                                              // 9°
                                                              // 10°

      try {
        Thread.sleep(50);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }}
```