Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina:
AP 1° semestre de 2007.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1)

Imagine que devemos implementar um jogo de guerra composto por canhões e aviões. Estes podem atirar, e cada acerto corresponde a 25% menos de vida do acertado, além de bonificar com 1 ponto o atirador. Tanto canhões quanto aviões possuem largura e altura próprias, de forma que um acerto corresponde a um tiro que atravessa a área destes objetos. Os aviões podem se mover de qualquer ponto para qualquer ponto da tela, enquanto que os canhões têm uma posição fixa. Entretanto, um canhão pode variar o ângulo de inclinação de sua arma de 0 a 180 graus.

a) **[1,5 pontos]** Declare as classes necessárias para este jogo, seus respectivos atributos e métodos, aplicando os conceitos de OO vistos, como herança, sempre que possível.

b) **[1,0 ponto]** De que maneira podemos obrigar que novos tipos de indivíduos (soldado, helicóptero, etc) a serem desenhados tenham, pelo menos, um método de desenho. Altere o código dado de maneira a incorporar esta obrigação.

Resposta: Nesta questão desejamos que os alunos coloquem em prática os conhecimentos passados de definição de classes, herança, etc. O que será cobrado é apenas a estruturação básica, como está no código abaixo. Ou seja, não é necessária a definição de um jogo completo. Para o item b, devemos definir um método de desenho abstrato, de forma que as especializações sejam obrigadas a definir este método.

```
abstract class ObjetoVisual {
	float vida; // Varia entre 0, .25, .5, 1
	int pontuacao; // + 1 para cada acerto
	int pos_x; // Posicao no eixo x
	int pos_y; // Posicao no eixo y
	float largura; // Largura do objeto
	float altura; // Altura do objeto

	abstract void desenha();
}

class Aviao extends ObjetoVisual {
	void movimenta (int x, int y) {
		pos_x = x;
		pos_y = y;
	}
	void desenha() {
		// Comandos para o desenho de um aviao
	}
}

class Canhao extends ObjetoVisual {
	float angulo; // 0 a 180
	void desenha() {
		// Comandos para o desenho de um canhao
	}
}

public class Jogo {
	ObjetoVisual [] objetos; // Conjunto de objetos a ser manipulado
pelo jogo
}
```

Questão 2)

[2,5 pontos] No código abaixo temos a declaração de 3 classes: uma classe para inteiros (Inteiro), outra para racionais (Racional), que estende a de inteiros, e uma classe para teste (Teste). Qual o valor impresso pelos 2 métodos *print* chamados pelo método *main* da classe Teste?

```
class Inteiro {
      int a;
      Inteiro (int x) {
            a = x;
      }
      void add (int x) {
            a = a + x;
      }
      void inc () {
            a = a + 1;
      }
      void print () {
            System.out.println(a);
      }
}

class Racional extends Inteiro {
      int b;
      Racional (int x, int y) {
            super(x); b = y;
      }
      void add (Racional x) {
            a = a * x.b + b * x.a;
            b = b * x.b;
      }
      void inc () {
            a = a + b;
      }
      void print () {
            System.out.print(a);
            System.out.print("/");
            System.out.println(b);
      }
}

class Teste
{
      public static void main(String[] a) {
            Inteiro i = new Inteiro(1);
            Racional r = new Racional(2,2);
            i = r; i.inc(); i.print();
            i.add(1); i.print();
      }
}
```

Resposta: Nesta questão objetivamos avaliar o entendimento do aluno com respeito a chamada de métodos polimórficos.

Inicialmente, os objetos i e r são alocados. O objeto i tem o campo a=1, enquanto que o objeto r tem os campos a=2 e b=2. Após i receber r (i=r;), i passa a apontar para a área de memória de r, e ser um objeto da classe Racional.

Assim, quando o método inc é aplicado ao objeto i, o método chamado é o da classe Racional, o qual faz com que o campo a passe a valer 4. Com isso, o primeiro print imprime o valor 4/2.

Após, o método add com argumento 1 (int) também é aplicado ao objeto i. Como o método add da classe Racional recebe um objeto racional, o método add que é acionado é o da superclasse (Inteiro), o que incrementa em a o argumento passado. Assim, o campo a do objeto i passa a ter o valor 5. Com isso, o segundo print imprime o valor 5/2.

Ou seja, as saídas do programa são 4/2 e 5/2.

Questão 3)

Analise o código da classe Produto e responda às questões que se seguem.

```
class Produto{
   static int próximo_id = 0;
   private int id;
   private String nome;

   Produto (String nome){
      this.nome = nome;
      this.id = ++próximo_id;
   }

   Produto (Produto outro){
      this.nome = outro.nome;
      this.id = outro.id;
   }

   public static void main (String[] args) {
      Produto p1=new Produto ("Banana");
      Produto p2=new Produto (p1);
      if (p1.equals(p2))
            System.out.println ("Produtos iguais");
      else
            System.out.println ("Produtos diferentes");
   }
}
```

a) **[1,0 ponto]** Após a criação de p1 e p2, quais são os identificadores (id) desses objetos?

Resposta:
p1.id = 1 e p2.id = 1

b) **[1,5 pontos]** Ao executar a classe Produto (java Produto), a mensagem Produtos diferentes é mostrada na console, porque o método equals que está sendo usado é o que foi herdado da classe Object. Neste caso a implementação define que seus parâmetros são iguais se eles forem referências a um mesmo objeto. Altere a classe Produto, sem modificar o método main, para que a mensagem "Produtos iguais" seja mostrada na console.

Resposta:

```
public boolean equals(Object obj){
  if ((id == ((Produto) obj).id) && (nome.equals(((Produto) obj).nome))
    return true;

  return false;
}
```

Questão 4)

a) **[1,5 pontos]** O que são polimorfismo e late binding? Analise o código abaixo e informe, neste código, onde eles são utilizados.

Resposta:
Polimorfismo: capacidade de um objeto adquirir diversas formas.
Exemplo: `emp[0] = new Gerente("Luiz", 10000, "Vanessa");`

Late binding (ou amarração tardia): é a capacidade de adiar a resolução de um método até o momento no qual ele deve ser efetivamente chamado.
Exemplo: `emp[0].imprimeDados();`

b) **[1,0 ponto]** O que é type casting? Dê um exemplo no código abaixo.

Resposta:
Type casting: é a conversão explícita de um objeto de um tipo para outro.
Exemplo: `System.out.println(((Gerente)geral).retornaNomeSecretaria());`

```
class Empregado{
  String nome;
  float sal;

  Empregado(String n, float s){ nome = n; sal = s; }
  void imprimeDados(){ System.out.println(nome + " " + sal); }
  void aumentaSalario(float t){ sal = sal + (sal * t) / 100; }
}

class Gerente extends Empregado{
  String sec;

  Gerente(String n, float s, String ns){ super(n,s); sec = ns; }

  void aumentaSalario(float t){
      super.aumentaSalario(t);
      sal = 1.1 * sal;
  }

  void imprimeDados(){
      System.out.println(nome + " " + sal + " " + sec);
  }

  String retornaNomeSecretaria(){ return sec; }
```

```
  public static void main(String[] args){
    Empregado[] emp = new Empregado[2];
    emp[0] = new Gerente("Luiz", 10000, "Vanessa");
    emp[1] = new Empregado("Jose", 500);
    Empregado geral = emp[0];

    if (geral instanceof Gerente)
      System.out.println(((Gerente)geral).retornaNomeSecretaria());

    emp[0].imprimeDados();
    emp[1].imprimeDados();
    geral.imprimeDados();
    emp[0].aumentaSalario(5);
    emp[1].aumentaSalario(8);
    geral = emp[1];
    System.out.println("Nova bateria de testes... Aumento");
    emp[0].imprimeDados();
    emp[1].imprimeDados();
    geral.imprimeDados();
  }
}
```