UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
SISTEMAS DE INFORMAÇÃO - CÂMPUS DE COXIM
FUNDAMENTOS EM ORIENTAÇÃO A OBJETOS

Data final de entrega 16/09/2014, até às 23h59min.
Enviar o arquivo de respostas em formato PDF e o arquivo.zip com códigos fontes para o e-mail mota.fernandomaia@gmail.com, insira no assunto do e-mail "[Lista 2 – Fundamentos em Orientação a Objetos]".

Lista de Exercícios – 02

1. Descreva com suas palavras a principal característica de um objeto, método ou argumento de função polimórfico.

2. Em relação ao código a seguir:

```
public class Janela {
    private Double largura;
    private Double altura;
    private String cor;
         .
         .
         .

}

public class JanelaQuarto extends Janela{
    private boolean persiana;
         private int numero;
         .
         .
         .

         public Double getArea(){
         return super.getAltura()*super.getLargura();
    }

    public Double getArea(Double areaMoldura){
         return super.getAltura()*super.getLargura()+areaMoldura;
    }
}
```

```
public class Main {
    public static void main(String args[]){
        JanelaQuarto janela = new JanelaQuarto();
        janela.getArea();
        janela.getArea(1.45);
    }
}
```

a. Descreva qual é o tipo de polimorfismo implementado.
b. É possível aplicar haver coerção neste exemplo? Se sim, por quê?

3. Em relação ao código a seguir:

```
public class Janela {
    private Double largura;
    private Double altura;
    private String cor;

        public Janela(){
                System.out.println("Olá, eu sou o construtor da classe Janela!");
                saida();
        }

    public void saida(){
        System.out.println("Para sair do quarto, utilize a porta frontal à janela!");
    }
}

public class JanelaQuarto extends Janela{
    private boolean persiana;
        private int numero;

        public JanelaQuarto(){
                System.out.println("Olá, eu sou o construtor da classe
JanelaQuarto!");
```

```
        }


        public void saida(){
        System.out.println("Para sair do quarto, utilize a porta lateral à janela!");
    }
}


public class Main {
    public static void main(String args[]){
                Janela janela = new Janela();
        System.out.println("");
        Janela janela2 = new JanelaQuarto();
    }
}
```

Sua Saída é:

```
Olá, eu sou o construtor da classe Janela!
Para sair do quarto, utilize a porta frontal à janela!

Olá, eu sou o construtor da classe Janela!
Para sair do quarto, utilize a porta lateral à janela!
Olá, eu sou o construtor da classe JanelaQuarto!
```

Descreva qual é a relação da ordem das mensagens de saída, em relação a polimorfismo e construtores.

4. Dentro da herança de classes é onde o polimorfismo de objetos é mais claro, desta forma baseado no código a seguir, responda:

    a. Em qual parte do código o polimorfismo fica explícito?
    b. Em qual momento o polimorfismo ocorre?

```
public class Janela {
    private Double largura;
    private Double altura;
    private String cor;
  .
  .
```

```
  .

  private void fechar(){
          //codigo omitido
  }
}

public class JanelaQuarto extends Janela{
    private boolean persiana;
  private int numero;
  .
  .
  .
  private void fechar(){
          if(persiana)
                  fecharPersiana();

          //codigo omitido
  }

  private void fecharPersiana(){
          //codigo omitido
  }
}

public class Main {
    public static void main(String args[]){
        Janela janela1 = new Janela();
        Janela janela2 = new JanelaQuarto();

          janela1.fechar();
          janela2.fechar();
    }
}
```

5. Defina encapsulamento com suas palavras.

6. Aponte e corrija o erros da classe *AponteErros*:

```
public class AponteErros {
    int id;
    String nome;
    Double velocidade;

    public aponteErros(int identificador, String nome){
        id=identificador;
        this.nome=nome;
    }

    private void setVelocidade(String v){
        velocidade=v;
    }

    private Double getVelocidade(){
        return velocidade;
    }
}
```

7. Descreva com suas palavras e através de exemplo como o encapsulamento pode facilitar tarefas de manutenção de software.

8. Decomponha o método *setLimpeza* da classe *Limpar*, de forma que reduza o acoplamento de dados existente.

```
public class Limpar {
    public void setLimpeza(String locais[]){
        int i;
        for(i=0;i < locais.length; i++){
            if(locais[i]=="quarto"){
                //codigo omitido
            }else if(locais[i]=="sala"){
                //codigo omitido
```

```
            }else if(locais[i]=="cozinha"){
                //codigo omitido
            }else{
                System.out.println("Informe um local válido!");
            }
        }
    }
}
```

9. Descreva as diferenças entre superclasse, classe abstrata e interface, utilize de exemplos de códigos para apresentar sua solução.

10. Programe as classes do diagrama de classes a seguir de acordo com as regras de POO.