**Основы программной инженерии. Вопросы и задания к зачету**

*1) Теоретические вопросы по программной инженерии*

1. Программная инженерия.
2. Жизненный цикл программного продукта.
3. Модели процесса разработки программного обеспечения.
4. ГОСТ 19, ГОСТ 34, SW-CMM
5. RUP, MSF
6. PSP/TSP, Agile
7. Структура процесса анализа требований.
8. Виды требований по уровням. Виды требований по характеру
9. Методы выявления требований.
10. Характеристики качественных требований.

*2) По коду построить диаграмму UML*

|  |  |
| --- | --- |
| 1 | type  TAbstractFigure = class  private  is\_visible: boolean;  public  procedure Show; virtual; abstract;  procedure Hide; virtual; abstract;  end;  TPixel = class(TAbstractFigure)  private  \_color: TColor;  \_point: TPoint;  protected  function GetX(): TCoord;  function GetY(): TCoord;  end; |
| 2 | type  TShape = class  procedure Show; virtual; abstract;  procedure Hide; virtual; abstract;  end;  TCircle = class(TShape)  procedure Show; override;  procedure Hide; override;  end;  TTriangle = class(TShape)  procedure Show; override;  procedure Hide; override;  procedure Rotate90; virtual;  end; |
| 3 | type  TLight = class  private  isTurned: boolean;  public  procedure TurnOn();  procedure TurnOff();  end;  TGarland = class  private  cl: array of TLight;  public  function GetLight(Index: integer): TLight;  function GetSize: integer;  end; |
| 4 | type  IColorable = interface  procedure SetColor(AColor: ColorEnum);  function GetColor: ColorEnum;  end;  TLight = class  private  isTurned: boolean;  public  procedure TurnOn();  procedure TurnOff();  function IsTurnedOn: boolean;  end;  TColoredLight = class(TLight, IColorable)  private  color: integer;  public  procedure SetColor(AColor: ColorEnum);  function GetColor: ColorEnum;  function ToString(): string;  end; |
| 5 | type  TUnit = class  public  constructor Create(ACode: word; AName: TName; AShortName: TName = '');  function GetName: TName;  procedure SetName(AName: TName);  procedure Print;  end;  TUnitList = class  private  \_items: array of TUnit;  public  constructor Create(AFileName: string);  procedure Load;  procedure Save;  end; |
| 6 | type  TIntegerValue = class(TObject)  private  value: integer;  public  function GetValue(): integer;  end;  TRatio = class(TObject)  private  n, d: TIntegerValue;  public  counter: integer; static;  constructor Create;  constructor Create(other: double);  end; |
| 7 | TCell = class(TObject)  private  FBorder: TBorder;  FValue: TString;  FRow: TRow;  FColumn: TColumn;  public  constructor Init(AValue: TString; ABorder: TBorder);  procedure Show(); virtual; abstract;  end;  TColumn = class(TObject)  Width: byte; // ширина столбца  end;  TRow = class(TObject)  public  procedure Draw(var ReportFile: text);  procedure SetCell(Index: Integer; ACell:TCell);  end; |
| 8 | TCell = class(TObject)  private  FBorder: TBorder;  FValue: TString;  public  procedure Show(); virtual; abstract;  end;  TCellBuilder = class  public  function CreateCell(): TCell; virtual; abstract;  end;  TAstericsCell = class(TCell)  public  procedure Show(); override;  end;  TAstericsCellBuilder = class(TCellBuilder)  public  function CreateCell(): TCell; override;  end; |
| 9 | TCell = class(TObject)  private  FColumn: TColumn;  public  procedure Show(); virtual; abstract;  end;  TColumn = class(TObject)  Caption: TString;  Dataset: TDataset;  end;  TDataset = class(TObject)  Columns: array of TColumn;  procedure Draw(var ReportFile: text);  function AddColumn(ACaption: TString; AAlign: TAlign; AWidth: byte): TColumn;  end; |
| 10 | TCell = class  private  FRow: TRow;  FColumn: TColumn;  end;  TCellBuilder = class  public  function CreateCell(): TCell; virtual; abstract;  end;  TDataset = class  Report: TReport;  end;  TReport = class(TObject)  public  Filename: TString;  Data: array of TDataset;  CellBuilder: TCellBuilder;  procedure Save();  end; |

*3) По диаграмме UML написать код*

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

*4) Вопросы по ООП. Объяснение проводить на конкретном примере (можно из своего варианта по лабораторным работам).*

1. Описание класса. Задание полей и методов. Реализация методов.
2. Конструкторы. Деструкторы. Инициализация объектов производного класса
3. Члены класса. Статические члены класса. Неявный параметр self.
4. Управление доступом к членам класса.
5. Переопределение методов в производном классе
6. Статическое связывание. Динамическое связывание.
7. Абстрактные методы. Абстрактные классы. Интерфейсы.
8. Проверка типа. Приведение типа.
9. Генерация исключения
10. Обработка исключений: try-except, try-finally. Определение типа исключений в блоке обработчике