*Logic Specification Template*

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | David Zavala López | **Program #** | 4 |

**Include the image of the activity diagram created for the program:**

Clase: Simpsons

|  |  |
| --- | --- |
| **Nombre** | **Parámetros** |
| Gamma | x: float >= .5 |
| If x = 1 -> return = 1  If x = 0.5 -> return = sqrt(pi)  return = (x-1) \* Gamma(x-1) | |
| TDistributionPDF | x: int >= 0  dof: int > 0 |
| numerator := Gamma((dof + 1) / 2)  denominator := (Gamma(dof / 2) \* ((dof \* pi) \*\* 0.5)))  return = (nominator / denominator) \* (1 + (x \*\* 2 / dof)) \*\* (-(dof + 1) / 2) | |
| SimpsonsRuleIntegration | func: método que represente una función aplicable para el caso  n: int > 1  W: float -> x/n  x : float >= 0 |
| sumOdds := GetSumOdds()  sumEvens := GetSumEvens()  integral := (W / 3) \* (func(0,dof) + sumOdds + sumEvens + func(x,dof))  return = integral | |
| GetSumOdds(func, n, W, dof) | func: método que represente una función aplicable para el caso  n: int > 1  W: float -> x/n  dof: int > 0 |
| sumOdds := 0  ∀ i ∈ oddNumbers : 0 .. n – 1 | sumOdds := sumOdds + (4 \* simpson(i \* W, dof))  return = sumOdds | |
| GetSumEvens(func, n, W, dof) | func: método que represente una función aplicable para el caso  n: int > 1  W: float -> x/n  dof: int > 0 |
| sumEvens := 0  ∀ i ∈ evenNumbers : 0 .. n – 1 | sumEvens := sumEvens + (2 \* simpson(i \* W, dof))  return = sumEvens | |
| IntegrateWithSimpsonsRule | x: float >= 0  dof: int > 0  n: int > 1 |
| E := .0000001  probability1 := SimpsonsRuleIntegration(TDistributionPDF, n, x/n, x, dof)  n := n\*2  probability2 := SimpsonsRuleIntegration(TDistributionPDF, n, x/n, x, dof)  if abs (probability2 – probability1) > E ->  probability1 = probability2;  n \* 2;  repeat;  return = probability2 | |
| CalculateIntegrateBySimpsonsRule(x, dof, n=10) | x: float >= 0  dof: int > 0  n: int > 1 |
| if fileExists(“Cache/”+str(x)+’-‘+str(dof)+’.txt’) ->  open(“Cache/”+str(x)+’-‘+str(dof)+’.txt’)  return = file  else ->  p := IntegrateWithSimpsonsRule(x, dof, n)  Imprimir resultados en el formato indicado  saveResultInCache(x,dof,p) | |

Clase: Main

|  |  |
| --- | --- |
| **Declaración** | **Parámetros** |
| main() |  |
| x := askForX()  dof := askForDof()  CalculateIntegrateBySimpsonsRule(x, dof) | |

Clase: FileHelper

|  |  |
| --- | --- |
| **Declaración** | **Parámetros** |
| saveResultsInCache(content,name) | content: string -> contenido del archivo a guardar  name: string -> nombre del archivo a guardar |
| Si name acaba con .txt:  path = “Cache/” y name  Si no:  path = “Cache/” y name y “.txt”  Crear archivo en path y escribir la variable contet en el | |

Clase: StringHelper

|  |  |
| --- | --- |
| **Declaración** | **Parámetros** |
| isNumber(string) | string: string |
| acceptedChars = ['0','1','2','3','4','5','6','7','8','9',',','.']  Iterar char sobre string:  Si char no esta en acceptedChars:  return = False  return = True | |
| removeCommas(string) | string: string |
| newValue = “”  Iterar char sobre string  Si char ¡= ‘,’:  Agregar char a newValue  return = newValue | |
| convertToFloat(string) | string: string |
| Intentar:  f := float(string)  return = f  Si no se pudo:  Avisar del error en pantalla  Regresar un valor nulo | |
| convertToInt(string) | string: string |
| Intentar:  f := int(string)  return = f  Si no se pudo:  Avisar del error en pantalla  Regresar un valor nulo | |

Clase: InputHelper

|  |  |
| --- | --- |
| **Declaración** | **Parámetros** |
| askForX() |  |
| Iteración infinita:  inputValue = input del usuario  Si inputValue está en blanco:  Volver a empezar la iteración  Si inputValue no es un número:  Volver a empezar la iteración  Remover comas de inputValue  inputAsNumber := Convertir inputValue en un float  Si inputAsNumber es nulo:  Volver a empezar la iteración  Si inputAsNumber < 0:  Volver a empezar la iteración    return = inputAsNumber | |
| askForDof() |  |
| Iteración infinita:  inputValue = input del usuario  Si inputValue está en blanco:  Volver a empezar la iteración  Si inputValue no es un número:  Volver a empezar la iteración  Remover comas de inputValue  inputAsNumber := Convertir inputValue en un int  Si inputAsNumber es nulo:  Volver a empezar la iteración  Si inputAsNumber < 0:  Volver a empezar la iteración    return = inputAsNumber | |