# Assignment Sheet

|  |  |
| --- | --- |
| **Course, instructor name & contact info** | Numeric Computing |
| **Assignment name** | Assignment 3 – Simple Interest Program |
| **Grade value** | 20%  Rubric attached |
| **Due date** | Week 11 |
| **Individual or group assignment** | Individual |
| **Submission instructions** | Show in class and submit on Blackboard |
| **Targeting these learning outcomes from course outline** | * produce computer programs that implement business mathematics. * Compare and contrast simple interest and compound interest in terms of the cost of borrowing money. * Compare and contrast the present and future value of an investment. * synthesize concepts and requirements for abstract algorithms to formulate solutions for specific business applications. |

Instructions:

* Create a Simple Interest program (simple text, no graphics or buttons) that uses the SimpleInterest class as described in the following UML diagram. Feel free to change to suit your style:

|  |
| --- |
| SimpleInterest |
| Member variables:  private double principal  private double rate  private double periodYears |
| Constructor and methods:  public SimpleInterest(double principalValue, double rateValue, double periodYearsValue)  public double getPrincipal( )  public double getRate()  public double getPeriodYears()  public void setPrincipal(double principalValue)  public void setRate(double rateValue)  public void setPeriodYears(double years)  public void setPeriodMonths(double months)  public void setPeriodDays(double days)  public double getSimpleInterest()  public double getMaturityValue()  @Overide public String toString() |
|  |

Test all your methods and your program with the following examples:

1. Simple interest for $1000 at 10% p.a. for 3 years = $300
2. Simple interest for $3200 at 6% p.a for 125 days = $65.75
3. The maturity value of $700 at an interest rate of 1.5% p.m. after 4 months = $742

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
|  | 5% | 4-3% | 2% | 1% | 0% |
| **Code quality** | Coded extremely well and wonderful comments | Coded to specifications | Missing or broken functionality | Missing lots of detail | Largely empty |
| **Test case** | Test case coded well and is clear. Console output is well formatted to see clearly code working as expected. | Coded to specifications | Missing or broken functionality | Missing lots of detail | Largely empty |
| **Presentation** | Able to explain code very well in person and handle live-coding easily | Satisfactory presentation of work done | Not able to explain all code or cannot make certain changes | Somewhat confusing or vague | Very confusing. Unsure student understands any code. Or no presentation |
| **Process** | Pieces of the assignment have been steadily building throughout the time given. Creativity used to solve key problems. Able to reflect on what they could improve on. | Assignment completed satisfactorily | Somewhat of a rush to complete assignment | Obviously has been a big last-minute rush to complete assignment. No time for creativity or reflection | Student has not spent any time grappling with concepts, reaching out for help, or reflecting on topic |

## Rubric

## Grading standard

20/20 - Work so amazing the instructor would only see this once in a lifetime

18/20 – Exceptional work, rare

16/20 - Great work, student has full command of the topic.

13/20 - Minor errors

10/20 - Errors and perhaps a major error

8/20 - Regular and consistent major errors. Lack of understanding

4/20 - Largely empty