King Saud University

College of Computer and Information Sciences

Computer Science Department

Course Code:	CSC 113
Course Title:	Computer Programming II
Semester:	Fall 2022
Exercises Cover Sheet:	**Final Exam**

Student Name:	
Student ID:	
Section No/ Time.	

Tick the Relevant	Computer Science B.Sc. Program ABET Student Outcomes	Question No. Relevant Is Hyperlinked	Covering%
	a) Apply knowledge of computing and mathematics appropriate to the computer science;		
	b) Analyze a problem, and identify and define the computing requirements appropriate to its solution		
	c) Design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;		
	d) Function effectively on teams to accomplish a common goal;		
	e) Understanding of professional, ethical, legal, security, and social issues and responsibilities;		
	f) Communicate effectively with a range of audiences;		
	g) Analyze the local and global impact of computing on individuals, organizations and society;		
	h) Recognition of the need for, and an ability to engage in, continuing professional development;		
	i) Use current techniques, skills, and tools necessary for computing practices.		
	j) Apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;		
	k) Apply design and development principles in the construction of software systems of varying complexity;		

Name :_____ **ID:**_____

EXERCISE 1: 20 * 1.25 each = 25

- There are **23** questions in **this exercise**. To get the full mark of this question, you **must** answer at **least 20** (out of 23) questions correctly.
- **No extra credits** if you answer all 23 questions correctly.
- Assume **no syntax/compilation errors** in coding questions.

Question	Answer
1.	A
2.	B
3.	B
4.	B
5.	A
6.	C
7.	B
8.	B
9.	D
10.	C
11.	B
12.	A
13.	D
14.	A
15.	C
16.	D
17.	A
18.	C
19.	A
20.	D
21.	D
22.	C
23.	C

Name :_____ **ID:**_____

1. A class `Student` and its subclass `GradStudent` both have a method `m1()`. If `st1` refers to an object of type `GradStudent`, what will `st1.m1()` do?

A. The `m1()` method defined in `GradStudent` will be called.
B. The `m1()` method defined in `Student` will be called.
C. Compiler will complain that `m1()` has been defined twice.
D. The `m1()` method defined in `GradStudent` will be called first then `m1()` method defined in `Student` will be called later.

2. Attributes, constructors, and methods can be inherited by its subclasses.
A. True
B. False

3. A protected variable is:
A. Accessible only within the class
B. Accessible only within the class and its subclass(es)
C. Accessible within the class and outside class
D. Accessible by all classes

4. A subclass constructor can invoke a superclass constructor by calling `super()` or `super(..)` at any statement.
A. True
B. False

5. A recursive binary search is an example of linear recursion.
A. True
B. False

Name :_____ **ID:**_____

6. What is the output of the following code:

```
class Food
{
  int items;
  int display( )
  {return items;}
}
```

```
public class Test
{
  public static void main(String[] args)
  {
    Food f = new Food();
    f.items = 4;
    System.out.println("Items Before = " + f.display( ));
    change(f);
    System.out.println("Items After = " + f.display( ));
  }
  static void change(Food foo)
  { foo.items = 8; }
}
```

A	B	C	D
Items Before = 8 Items After = 8	Items Before = 4 Items After = 4	Items Before = 4 Items After = 8	Items Before = 8 Items After = 4

7. Which of these statements is NOT correct?

A. A try block can have multiple catch blocks
B. A try block can have multiple finally blocks
C. A try block can contain try and catch blocks
D. A catch block can contain try and catch blocks

8. What is the output of the following code:

```
int a[] = new int[2];
try {
    a[0] = 3 / 2;
    a[1] = 0 / 3;
    a[2] = 4 / 5;
} catch (ArithmeticException e) {
    System.out.println("This is Arithmetic Exception");
} catch (Exception e) {
    System.out.println("This is Exception");
}
```

A. This is Arithmetic Exception
B. This is Exception
C. A and B
D. Runtime error

Name :_____ **ID:**_____

9. What is the output of the following code:

```
    public static void main(String
args[]) throws Exception {
        int a[] = {1, 2, 3};
        m1(a, 0);
    }
```

```
   public static void m1
(int a[], int x) throws Exception {

    if (x == a.length - 1) {
            throw new Exception();
      }
       System.out.print(a[x] + ",");
       m1(a, x + 1);

    }
```

A. 1,2,3
B. 1,2
C. 1
D. Runtime error

10. Choose the correct one for the following recursive method when n is 3 :

```
int recursiveSum(int n)
{
     if(n==0)
         return 0;
     return n + recursiveSum(n-1);
}
```

A. Every recursive call shares the same copy of parameter ***n*** in memory.
B. First and Last recursive call share the same copy of parameter ***n*** in memory.
C. There will be a separate copy of parameter ***n*** in memory for each recursive call.
D. Only First and Last recursive call have separate copies of parameter ***n*** in memory.

11. Suppose A is an abstract class, B is a concrete subclass of A, and both A and B have a default constructor. Which of the following is correct?

A. `A a = new A(); and A a = new B();`
B. `A a = new B(); and B b = new B();`
C. `B b = new A(); and B b = new B();`
D. `A a = new A(); and B b = new A();`

Name :_____ **ID:**_____

12. What happens during execution if a negative value is used for an array index?

A. An `IndexOutOfBoundsException` is thrown.

B. A `NumberFormatException` is thrown.

C. The first slot of the array is used.

D. Compilation Error.

13. What is the output of the following code:

```
class exception_handling
{
      public static void main(String args[])
      {
            try
            {
                  throw new NullPointerException ("A");
            }
            catch(ArithmeticException e)
            {
                  System.out.print("B");
            }
      }
}
```

A. A

B. B

C. AB

D. Runtime Error

14. Which of these is a super class of all exception classes?

A. Exception
B. RuntimeError
C. Throw
D. Catch

Name :_____ **ID:**_____

15. What will be the output of the code:

```
public static void main(String args[])
     {
          try
          {
               int sum = 1 / 0;
               System.out.print("Sum is " + sum);
          }
          catch(ArithmeticException e)
          {
               System.out.println("Exception");
          }
     }
}
```

A. Sum is 0
B. Sum is 0 Exception
C. Exception
D. Sum is Exception

16. Given the following function, what will `stars(3)` return?

```
public String stars(int n)
{
   if (n == 0) {
      return "*";
   } else {
      return stars(n - 1) + stars(n - 1) + " ";
   }
}
```

A. * *
B. ** **
C. **** ****
D. ** ** ** **

Name :_____ **ID:**_____

17. The class `FileInputStream` is used to read ___ from a binary file

A. Bytes
B. Characters
C. String
D. Numbers

18. What is the class used to write double data to a binary file using the method "`writeDouble()`"?

A. ObjectOutputStream
B. DataInputStream
C. DataOutputStream
D. FileOutputStream

19. To read the data from a file correctly:

A. We must know the order of the data stored and their data types
B. We must know the order of the data stored only
C. We must know the data types only
D. We do not need to know the order of the data stored and their data types

20. Which of the following may be considered as an advantage of the array structure?

A. number of cells is fixed
B. waste of memory space may happen if the number of data inserted in the array are too little compared to the size of the array
C. adding is made specified position without shifting
D. visiting data at specified cell number

21. Let's consider a linked list implementation with one reference of node head. Which of following will change the head in an EMPTY linked list?

A. insertAtFront
B. insertAtBack
C. removeFromFront
D. A and B

Name :_____ **ID:**_____

22. What does the following method do for a given Linked List of integers

```
static void fun(Node head)
{
     if (head == null)
     {
          return;
     }

     fun(head.next);
     System.out.print(head.data + " ");
}
```

A. Prints all nodes of linked lists

B. Prints alternate nodes of Linked List

C. Prints all nodes of linked list in reverse order

D. Prints alternate nodes in reverse order

23. Assume `NumList` is a LinkedList that contains integers {1,2,3,4,5}. The value of x after the execution of this code:

```
public class Node {
     public int data;
     public Node next;
     ...
}
```

```
public class LinkedList {
     public Node head;
     ...
}
```

```
int x = 0;
Node current = NumList.head;
while (current.next != null) {
    if (current.data >= 2)
        x++;
    current = current.next;
}
```

A. 1
B. 2
C. 3
D. 4

Name :_____ **ID:**_____

EXERCISE 2: Write the output of this program. Assume no syntax/compilation errors

5 marks

```
public class Engine {

    private boolean on = true;

    public void start() throws Exception {
        if (on) {
            throw new Exception("Wrong start action");
        }
        System.out.println("Engine Started..");

    }

    public void stop() throws Exception {
        if (!on) {
            throw new Exception("Wrong stop action");
        }
        on = false;
        System.out.println("Engine Stopped..");

    }
}
public class Car {

    private String color = "NoColor";
    private int mileage;

    public Car() throws Exception {
        this("White", 0);
    }

    public Car(String color, int mileage) throws Exception {

        if (mileage < 0 || mileage > 50) {
            throw new Exception("Wrong value");
        }
        this.color = color;
        this.mileage = mileage;
        print();
    }

    public void print() {
        System.out.println("Car Color= " + color
                + " Mileage= " + mileage);
```

Name :_____ ID:_____

```
    }

}

public class BMW extends Car {

    private Engine BMWEngine;

    public BMW() throws Exception {
        BMWEngine = new Engine();
        BMWDemo();

    }

    public BMW(String color, int mileage) throws Exception {
        super(color, mileage);
        BMWEngine = new Engine();
        try {
            BMWEngine.stop();
            BMWDemo();
        } catch (Exception e) {
            System.out.println("BMW Con Exception");

        }
    }

    public void BMWDemo() throws Exception {
        print();
        BMWEngine.start();
        BMWEngine.stop();

        try {
            BMWEngine.start();

        } catch (Exception e) {
            System.out.println("BMWDemo Exception");
        }

    }

}
```

Name :_____ **ID:**_____

```
public class main {

    public static void main(String[] args) {
        String colors[] = {"Red", "Black", "White", "Blue"};
        int mileage[] = {15, 40, 120, 20};
        int j = 0;
        Car myCars[] = new Car[4];

        for (int i = 0; i <= 4; i++) {
            try {
                if (i % 2 == 0) {
                    myCars[j++] = new BMW(colors[i], mileage[i]);
                } else {
                    myCars[j++] = new Car(colors[i], mileage[i]);
                }
            } catch (ArrayIndexOutOfBoundsException e) {
                System.out.println("Wrong index: " + i);
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
        System.out.println("The number of inserted cars is: " + j);
    }
}
```

Name :_____ **ID:**_____

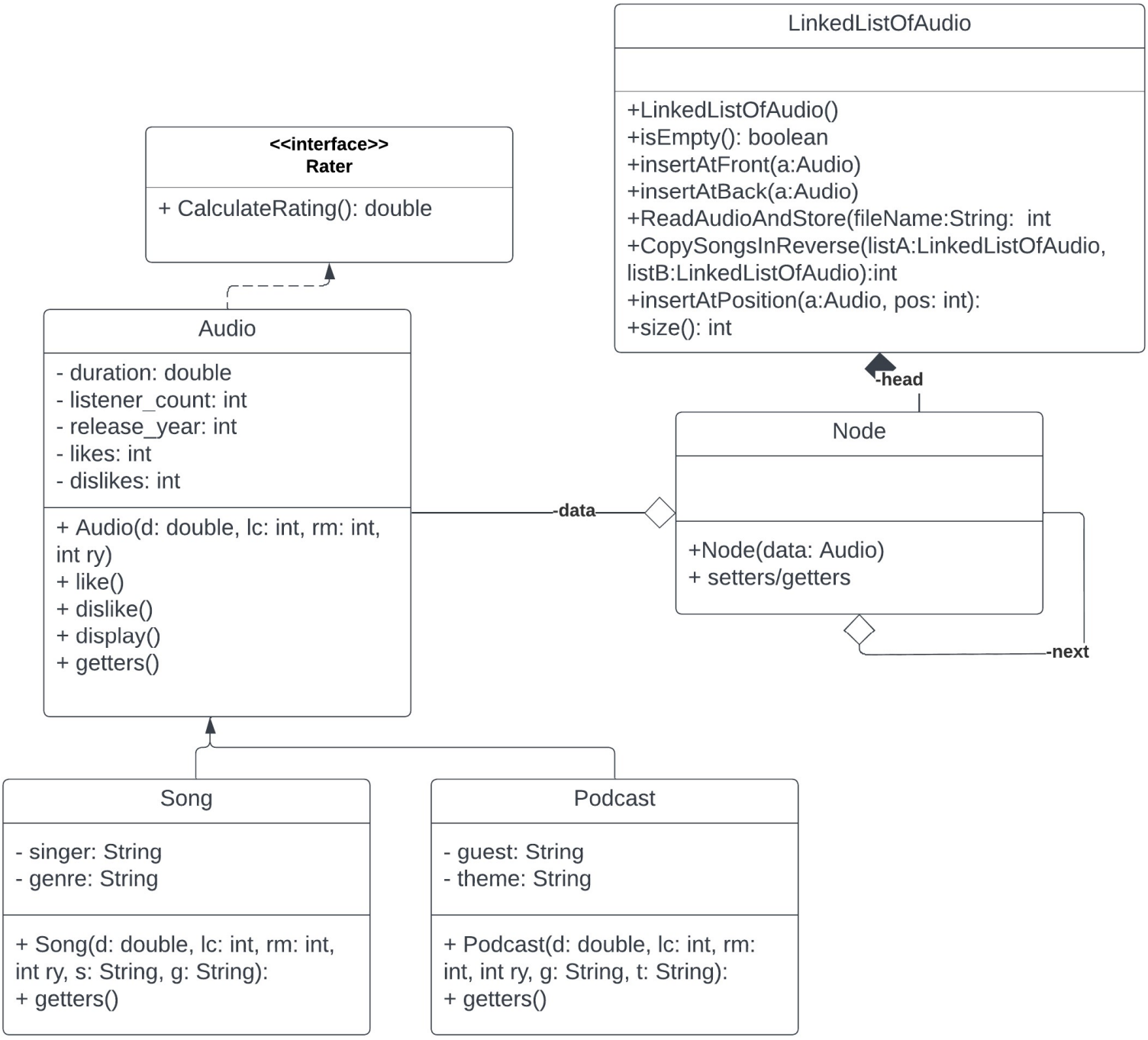
Answer: (0.5 each line)

```
Car Color= Red Mileage= 15
Engine Stopped..
Car Color= Red Mileage= 15
Engine Started..
BMW Con Exception
Car Color= Black Mileage= 40
Wrong value
Car Color= Blue Mileage= 20
Wrong index: 4
The number of inserted cars is: 5
```

EXERCISE 3: 10 marks

Consider the following UML class diagram:

Name :_____ **ID:**_____



Interface Class ***Rater:***

- Methods:

- ***CalculateRating():*** this method calculate the rating of the audio based on its type as follows:
  - ***Song*** returns the rating based on the following formula:
    - ***(likes/ (likes+ dislikes)) * 100***
  - ***Podcast*** returns the rating based on the following formula:
    - ***(likes/ (likes+ (dislikes*1.2))) * 100***

Class ***Audio:***

- Attributes:
  - **duration:** the duration of an audio in seconds.
  - **listener_count*:*** number of lisinter for the audio.
  - ***release_year:*** the year of release date.
  - ***likes:*** number of likes for the audio.
  - ***dislikes:*** number of dislikes for the audio.

- Methods:
  - ***Audio(d: double, lc: int, rm: int, int ry) : constructor***.
  - ***like():*** increase the number of likes by 1.
  - ***dislike():*** increase the number of dislikes by 1.
  - ***display():*** prints all the attributes of the audio.
  - ***getters() :*** return the value of each attribute.

Class ***Song:***

- Attributes:
  - ***singer:*** the name of the singer.
  - ***genre***: category of the song
- Methods:
  - ***Song(d: double, lc: int, rm: int, int ry, s: String, g: String): constructor.***
  - ***getters() :*** return the value of each attribute.

Class ***Podcast:***

- Attributes:
  - ***guest:*** the name of the guest for the podcast.
  - ***theme:*** category of the podcast
- Methods:
  - ***Podcast(d: double, lc: int, rm: int, int ry, g: String, t: String): constructor***.
  - ***getters() :*** return the value of each attribute.

Class ***LinkedListOfAudio***:

- Methods:

Name :_____ **ID:**_____

- **LinkedListOf*Audio* (…):** Constructor**.**
- **isEmpty():** This method will return true if the list is empty.
- **insertAtFront(…):** This method will insert a new node at the beginning of the list**.**
- **insertAtBack(…):** This method will insert a new node at the end of the list.
- **ReadAudioAndStore(String:fileName):** This method will read and store all audio from file fileName at the back of the list. This method should return how many audios you have added.
- **CopySongsInReverse(LinkedListOfAudio:listA, LinkedListOfAudio:listB):** This method will receive non-empty list listA and an empty list listB. Write all songs (in reverse order) with duration more than 600 seconds from listA to listB. You should return how many songs you have added in listB.
- **insertAtPosition(a: Audio, pos: int):** This method will add the audio **a** at position (index) pos. If **pos** value is 0, then the audio **a** should be inserted at the beginning of the list. If **pos** is larger than the size of the list, then audio **a** should be inserted at the end of the list. If **pos** is negative, then this method should raise an Exception with the following message "negative index!".
- **size( ):** this method will return the size of the list.

Complete the following methods:

```
public int ReadAudioAndStore(String FileName) {
```

3 marks

Name :_____ **ID:**_____

```
File f = new File(fileName); 0.25
FileInputStream fis = new FileInputStream(f); 0.25
ObjectInputStream ois = new ObjectInputStream(fis); 0.25
int counter = 0; 0.25

try {0.25

    while (true) { 0.25

        Audio s1 = (Audio) ois.readObject(); 0.5
        insertAtBack(s1); 0.5
        counter++;0.25

    }

} catch (ClassNotFoundException ex) {
    System.out.println("ClassNotFoundException ");
} catch (EOFException e) { 0.25 // or Exception
    System.out.println("Reading from file is Done!!");
}

ois.close();
fis.close();
return counter;
```

```
}
```

Name :_____ **ID:**_____

```
public int CopySongsInReverse(LinkedListOfAudio listA,
LinkedListOfAudio listB) {
```

3 marks

```
        Node current = head; 0.25
        while (current!= null) { 0.5
            if (current.data instanceof Song &&
current.data.duration > 600) { 0.5  0.5
                listB.insertAtFront(current.data); 0.5


            }

            current = current.next; 0.5


        }

        return listB.size();0.25




}
```

```
public void insertAtPosition(Audio a, int pos) throws Exception{
```

4 marks

Name :_____ **ID:**_____

```
        if (pos < 0) {0.25
            throw new Exception("Wrong index"); 0.50
        } else if (pos == 0) { 0.25
            insertAtFront(a); 0.25
        } else if (pos >= size()) {0.25
            insertAtBack(a); 0.25
        } else {

            int counter = 0; 0.25
            Node current = head; 0.25
            Node n = new Node(a); 0.25
            while (current != null) { 0.25
                counter++;0.25
                if (counter == pos) {0.25
                    n.next = current.next; 0.25
                    current.next = n; 0.25
                    break;
                } else {
                    current = current.next; 0.25
                }
            }

        }

}
```