OOM Assignment (Miscellaneous)

Name: Aditya Aggarwal **Roll Number: IIT2019210**

Q.1 Predict and justify the output of the following programs

a) **Output:** Derived::show() called

Justification: Since, b is a reference of Base type and refers to an object of Derived class. In Java, functions are virtual by default. So the run time polymorphism happens and derived show() is called.

b) **Output**: It will show error as : Compile error

Justification: Because in the show() method in parent class is final and can't be over ridden in the child class.

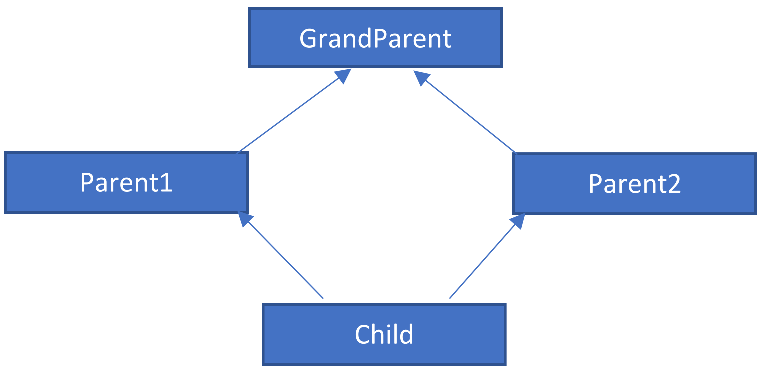
c) **Output:** Base::show() called

Justification: Due to Upcasting, over ridden show() should be called but When a function is static, runtime polymorphism doesn't happen. So, show() from Base would be called.

Q.2 In multiple inheritance one class inherits the properties of multiple classes. In other words, in multiple inheritance we can have one child class and n number of parent classes. Java does not support multiple inheritance (with classes). Using reference to the Diamond problem justify the above statement and elaborate by giving an example how we can implement multiple inheritance in Java on our own.

Solution:

In diamond problem, there are 4 classes: GrandParent, Parent1, Parent2 and Child. GrandParent is parent of Parent1 and Parent2, and Child is child class of Parent1 and Parent2.



This is called as diamond Problem because the class diagram has a diamond structure.

Code:

```
class GrandParent
{
   void fun()
   {
      System.out.println("Grandparent");
   }
}

class Parent1 extends GrandParent
{
   void fun()
   {
      System.out.println("Parent1");
   }
}

class Parent2 extends GrandParent
{
   void fun()
   {
      System.out.println("Parent2");
   }
}


class Test extends Parent1, Parent2
{
  public static void main(String args[])
  {
     Test t = new Test();
     t.fun();
  }
}
```

The above Java program throws compiler error when run. Multiple inheritance causes this diamond problem and hence multiple inheritance is not supported in java so it shows error on compiling it.

Instead, we can use interface to implement multiple inheritance in java.

Code:

```
interface GrandParent
{
   default void show()
```

```
    {
        System.out.println("Default Grand Parent");
    }
}

interface Parent1 extends GrandParent { }

interface Parent2 extends GrandParent { }

class TestClass implements Parent1, ParentI2
{
    public static void main(String args[])
    {
        TestClass d = new TestClass();
        d.show();
    }
}
```

Output: Default Grand Parent

Since a class can implement any number of interfaces so in order to implement multiple inheritance, interfaces are used.

Q.3 a) What is method overloading and method overriding? Also give examples depicting the differences.

Method Overloading is **compile time polymorphism**. In method overloading, more than one method have same name with different signature in the class. In this, the return type can or cannot be same, but we must have to change the parameter in signature.

Code:

```
Class TestClass{
  static int add(int a, int b){return a+b;}
  static int add(int a, int b, int c){return a+b+c;}      //Overloaded
method

   public static void main(String args[]) {
     System.out.println(add(4, 5));
     System.out.println(add(4, 5, 2));
   }
}
```

Output: 9

11

Method Overriding is a **Run time polymorphism**. In method overriding, derived class provides the some specific implementation of the over ridden method that is already provided by the base class or parent class. In method overriding, return type must be same.

Code:

```
class Animal{
      void eat(){System.out.println("eating.");}
      }
    class Dog extends Animal{
    void eat(){System.out.println("Dog is eating.");}
     }
   class TestClass{
    public static void main(String args[]) {
      Dog dog=new Dog();
      Animal ani=new Animal();
      dog.eat();
      ani.eat();
    }
}
```

Output: Dog is eating.

Eating

Method Overloading	**Method Overridding**
Method overloading is a compile time polymorphism.	Method overriding is a run time polymorphism.
Helps to rise the readability of the program.	Used to grant the specific implementation of the method which is already provided by its parent class.

Can be done within the class.	Can be done only in two classes with inheritance relationship
It may or may not require inheritance.	It always needs inheritance
In this, methods must have same name and different signature.	In this, methods must have same name and same signature.
Return type may or may not be same.	Return type must be same.

b) Why are pointers not used in Java?

Pointers are not used in java because of the following reasons:

1. Memory access via pointer arithmetic: Java has a robust security model and it doesn't pointer arithmetic for the same reason. It would be not possible for the Virtual Machine to ensure that code containing pointer arithmetic is safe without expensive runtime checks.

2. Security: By not allowing pointers, Java effectively provides another level of abstraction to the developer. No pointer support makes Java more secure because they point to memory location or used for memory management that loses the security as we use them directly.

3. Passing argument by reference: Passing a reference which allows you to change the value of a variable in the caller's scope. Java doesn't have this, but it's a pretty rare use case and can easily be done in other ways. This is in general equivalent to changing a field in an object scope that both the caller and callee can see.

4. Manual memory management: We can use pointers to manually control and allocate blocks of memory. This is useful for some bigger applications like games, device drivers etc. but for general-purpose Object-Oriented programming it is simply not worth the effort. Java instead provides very good automatic

Garbage Collection (GC) which takes care of memory management.

c) What is the difference between an Array list and vector in Java?

ArrayList	**Vector**
ArrayList is not synchronized.	Vector is synchronized.
ArrayList increments 50% of current array size (means 1.5 times the current size) if the number of elements exceeds from its capacity	Vector increments 100% of (means doubles the size) current array size if the total number of elements exceeds than its capacity.
ArrayList is not a legacy class. It is introduced in JDK 1.2.	Vector is a legacy class.
ArrayList is fast because it is non-synchronized	Vector is slow because it is synchronized, i.e., in a multithreading environment, it holds the other threads in runnable or non-runnable state until current thread releases the lock of the object.
ArrayList uses the Iterator interface to traverse the elements.	A Vector can use the Iterator interface or Enumeration interface to traverse the elements

d) What is singleton class in Java and how can we make a class singleton?

A singleton class is a class that can have only one object at a time i.e. only one instance of the class at the run time. After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. This means that whatever modifications we do to any variable inside the class through any instance, it will also affect the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.

In order to design a singleton class:

1. We can make the constructor as private.
2. We can write a static method that has return type object of this singleton class.

Q.4 a) What is the difference between equals() and == in Java?

Equal()	**==**
It is a method	It is an operator
It is use for content comparison	It is use for reference comparison

b) What are access modifiers in Java? Elaborate by making a table telling about the places in code where one can access objects/functions defined using each of the specifiers.

Access modifiers in Java helps to restrict the scope of a class, constructor, variable, method or data member. There are four types of access modifiers; public, protected, default and private.

Places\ Access Modifiers	public	protected	default	private
Same Class	Yes	Yes	Yes	Yes
Same Package sub class	Yes	Yes	Yes	No
Same Package non - sub class	Yes	Yes	Yes	No
Different Package sub class	Yes	Yes	No	No
Different Package non - sub class	Yes	No	No	No

Q.5 What is Java String Pool? Give an example.

Solution: String Pool is a storage area in Java heap. String Pool in Java is a pool of Strings stored in Java Heap Memory. We can create String objects using a new operator as well as providing values in double-quotes.

When we create a string like –

String str = “Abcde”

JVM automatically checks if the same value exists in the string constant pool or not, if exist then it occupies the already existing value, else it creates a new string by itself and adds it to the string pool.

If we want to halt this behaviour, create a string using new operator:

String newstr = new String(“Abcde”)

For example: String s1 = "Abcde";

String s2 = "Abcde";

String s3 = new String("Abcde");

String s4 = new String("Abcde").intern();

In this s1 = s2 = s4 ≠ s3

Q.6 Create a class named 'Member' having the following members:

Data members: Name, Age, Phone Number, Address, Salary It also has a method named 'printSalary' which prints the salary of the members.

Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

Solution: I have created 4 classes for this: Member, Employee, Manager and TestClass. Member has all given data members and then Employee and Manager are child of Member has have one data member more than Member. Finally TestClass is used to show the working of these 3 classes. Codes are given below:

Member:

```
package q6;

public class Member {
    String name, phoneNumber, address;
    int age;
    float salary;
    public Member(String Name, String number, String Address, int
Age, float Salary){
        this.name = Name;
        this.salary = Salary;
        this.address = Address;
        this.age = Age;
        this.phoneNumber = number;
    }

    public void printSalary(){
        System.out.println("Salary is: " + this.salary);
    }
```

```
        public void printDetails(String x){
            System.out.println("Name : "+ this.name);
            System.out.println("Age : "+ this.age);
            System.out.println("Phone Number : "+ this.phoneNumber);
            System.out.println("Address : "+ this.address);
            printSalary();
        }
    }
```

Employee:

```
    package q6;
    import q6.Member;
    public class Employee extends Member {
        String specialization;
        public Employee(String Name, String number, String Address, int
    Age, float Salary){
            super(Name, number, Address, Age, Salary);
        }

        void printDetails(){
            System.out.println("\nDetails of the Employee : ");
            super.printDetails("Employee");
        }
    }
```

Manager:

```
    package q6;
    import q6.Member;
    public class Manager extends Member {
        String department;
        public Manager(String Name, String number, String Address, int
    Age, float Salary){
            super(Name, number, Address, Age, Salary);
        }

        void printDetails(){
            System.out.println("\nDetails of the Manager : ");
            super.printDetails("Manager");
        }
    }
```

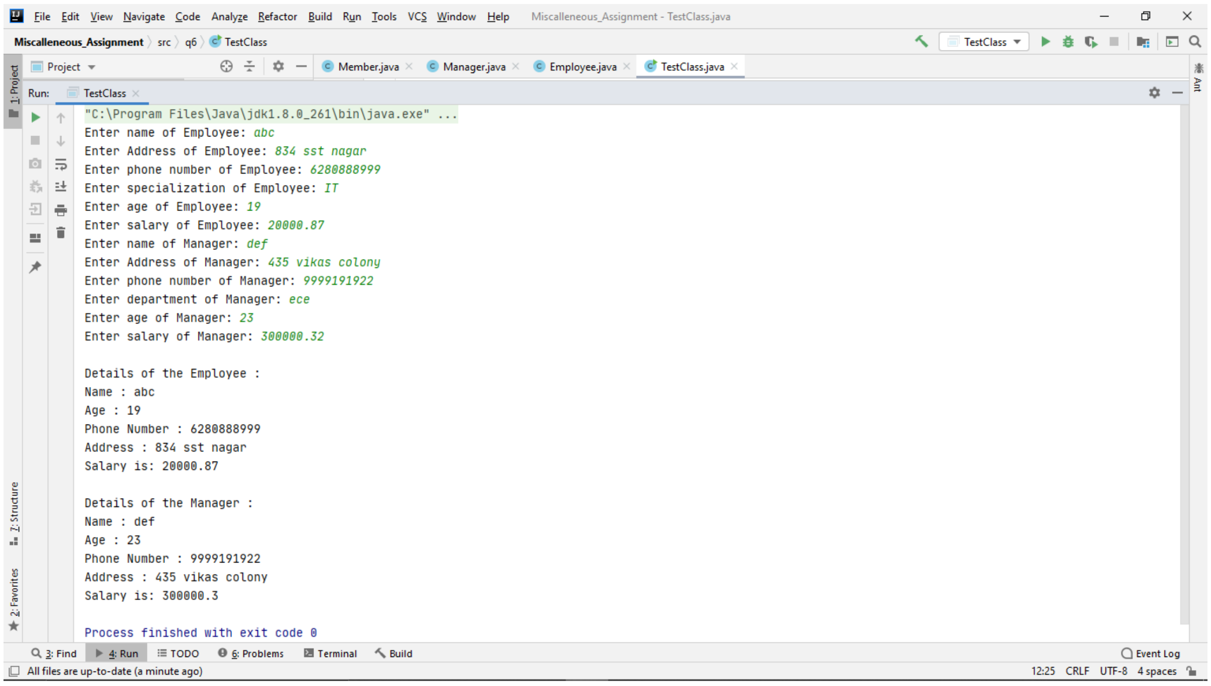
TestClass:

```
package q6;
import q6.*;

import java.util.Scanner;

public class TestClass {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String name,address,number,department,specialization;
        int age;
        float salary;
        //For Employee object
        System.out.printf("Enter name of Employee: ");
        name = s.nextLine();
        System.out.printf("Enter Address of Employee: ");
        address = s.nextLine();
        System.out.printf("Enter phone number of Employee: ");
        number = s.nextLine();
        System.out.printf("Enter specialization of Employee: ");
        specialization = s.nextLine();
        System.out.printf("Enter age of Employee: ");
        age = s.nextInt();
        System.out.printf("Enter salary of Employee: ");
        salary = s.nextFloat();
        //Creating object of Employee type
        Employee emp = new Employee(name, number, address, age, salary);
        emp.specialization = specialization;
        //Employee created with given details

        //For manager object
        System.out.printf("Enter name of Manager: ");
        name = s.nextLine();        //To ignore the next line error in java
scanner
        name = s.nextLine();
        System.out.printf("Enter Address of Manager: ");
        address = s.nextLine();
        System.out.printf("Enter phone number of Manager: ");
        number = s.nextLine();
        System.out.printf("Enter department of Manager: ");
        department = s.nextLine();
        System.out.printf("Enter age of Manager: ");
        age = s.nextInt();
        System.out.printf("Enter salary of Manager: ");
        salary = s.nextFloat();
        //Creating object of Manager type
        Manager man = new Manager(name, number, address, age, salary);
        man.department = department;
        //Manager created with given details

        //Now we can access them using objects emp and man

        //For employee
        emp.printDetails();

        //For manager
        man.printDetails();
    }
}
```

Sample Output:

```
Enter name of Employee: abc
Enter Address of Employee: 834 sst nagar
Enter phone number of Employee: 6280888999
Enter specialization of Employee: IT
Enter age of Employee: 19
Enter salary of Employee: 20000.87
Enter name of Manager: def
Enter Address of Manager: 435 vikas colony
Enter phone number of Manager: 9999191922
Enter department of Manager: ece
Enter age of Manager: 23
Enter salary of Manager: 300000.32

Details of the Employee :
Name : abc
Age : 19
Phone Number : 6280888999
Address : 834 sst nagar
Salary is: 20000.87

Details of the Manager :
Name : def
Age : 23
Phone Number : 9999191922
Address : 435 vikas colony
Salary is: 300000.3
```

Sample Screenshot:



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Enter name of Employee: abc
Enter Address of Employee: 834 sst nagar
Enter phone number of Employee: 6280888999
Enter specialization of Employee: IT
Enter age of Employee: 19
Enter salary of Employee: 20000.87
Enter name of Manager: def
Enter Address of Manager: 435 vikas colony
Enter phone number of Manager: 9999191922
Enter department of Manager: ece
Enter age of Manager: 23
Enter salary of Manager: 300000.32

Details of the Employee :
Name : abc
Age : 19
Phone Number : 6280888999
Address : 834 sst nagar
Salary is: 20000.87

Details of the Manager :
Name : def
Age : 23
Phone Number : 9999191922
Address : 435 vikas colony
Salary is: 300000.3

Process finished with exit code 0
```

I have also attached the java files of these 4 classes.

Q.7

Below is the UML representation of a set of classes that represent items that are used to compute a university's course grade. Your task is to implement this set of classes.

The line from Lab to Assignment indicates that a Lab is-a Assignment. You should use inheritance to capture this relationship. Likewise, for the Project and Assignment classes.

The line from AssignmentList to Assignment indicates that AssignmentList has-a Assignment.

Observe that there is an itemList attribute (instance variable) in the middle section of the

AssignmentList box. Other than including this attribute, there is nothing special to be done for a

has-a relationship. The "*" on this relationship indicates that the AssignmentList can have any

number of Assignment objects. This fact is captured in the AssignmentList through the use of

the ArrayList of Assignment (this is the meaning of the ArrayList notation in the type definition).

How much a particular Assignment contributes to your course grade is determined by three

properties: 1. score is the number of points that you have received on the assignment, 2.

totalPoints is the number of points that are possible for the assignment, and 3. totalWeight is the

weight of this Assignment relative to other assignments. AssignmentList.computeCourseGrade()

returns your score for the class based on the list of assignments. In particular:

where n is the total number of Assignments in the list, and total Weight[i] refers to the total

Weight of the ith Assignment. If n = 0, then the returned course grade must be zero.

Solution: Codes are given below:

Assignment:

```
package q7;

public class Assignment {
    String name;
    int month, day, hour, minute;
    double score, totalPoints, totalWeight;

    public Assignment(String name, int month, int day, int hour,
int min) {
        this.name = name;
        this.month = month;
        this.hour = hour;
        this.day = day;
        this.minute = min;
    }
    void setScore(double score){
        this.score = score;
    }
    void setTotalPoints(double tp){
        this.totalPoints = tp;
    }
    void setTotalWeight(double tw){
        this.totalWeight = tw;
    }
    String getName(){
        return name;
    }
    int getMonth(){
        return month;
    }
    int getDay(){
        return day;
    }
    int getHour(){
        return hour;
    }
    int getMinute(){
        return minute;
    }
    double getScore(){
        return score;
    }
    double getTotalPoints(){
        return totalPoints;
    }
    double getTotalWeight(){
        return totalWeight;
    }
    public String toString() {
        return "Assignment {" +
                "Name='" + getName()+ '\'' +
                ",\n Month=" + getMonth() +
                ",\n Day=" + getDay() +
                ",\n Hour=" + getHour() +
                ",\n Minute=" + getMinute() +
                ",\n Score=" + getScore() +
                ",\n Total Points=" + getTotalPoints() +
                ",\n Total Weight=" + getTotalWeight() +
                '}';
    }
}
```

AssignmentList:

```java
package q7;

import java.util.ArrayList;

public class AssignmentList {
    ArrayList<Assignment> listItems;

    public AssignmentList(){
        System.out.println("Creating new Assignment List");
        listItems = new ArrayList<Assignment>();
    }

    void addItem(Assignment item){
        listItems.add(item);
    }

    double computeCourseGrade(){
        double grade = 0.0;      //initialize to 0
        double sum = 0.0;        //initialize to 0
        int n = listItems.size();   //taking size of list
        for(int i=0;i<n;++i){
            double Score = listItems.get(i).score;
            double TotalPoints = listItems.get(i).totalPoints;
            double TotalWeight = listItems.get(i).totalWeight;
            sum += TotalWeight;
            grade += (TotalWeight*Score)/TotalPoints;
        }
        if(sum != 0.0)
            grade = (grade/sum) * 10;
        return grade;
    }
}
```

Lab:

```java
package q7;
import q7.Assignment;

public class Lab extends Assignment {
    String specification;
    Lab(String name,int month,int day, int hour,int minute, String
spec){
        super(name, month,day, hour,minute);
        this.specification = spec;
    }

    public String toString(){
        return "\nLab {\n" +

super.toString().substring(12,super.toString().length()-1)+
//Starting index is 12 because we want string from this index upto
last index
                ",\n Specification='" + this.specification + '\'' +
                "\n}";
    }
}
```

Project:

```
package q7;
import q7.Assignment;

public class Project extends Assignment {
    String specification;
    String dataFile;

    Project(String name,int month,int day, int hour,int minute,
String spec, String dataFile){
        super(name, month,day, hour,minute);
        this.specification = spec;
        this.dataFile = dataFile;
    }

    public String toString() {
        return "\nProject {\n" +

super.toString().substring(12,super.toString().length()-1)+
                ",\nSpecification='" + specification + '\'' +
                ",\nData File='" + dataFile + '\'' +
                "\n}";
    }
}
```

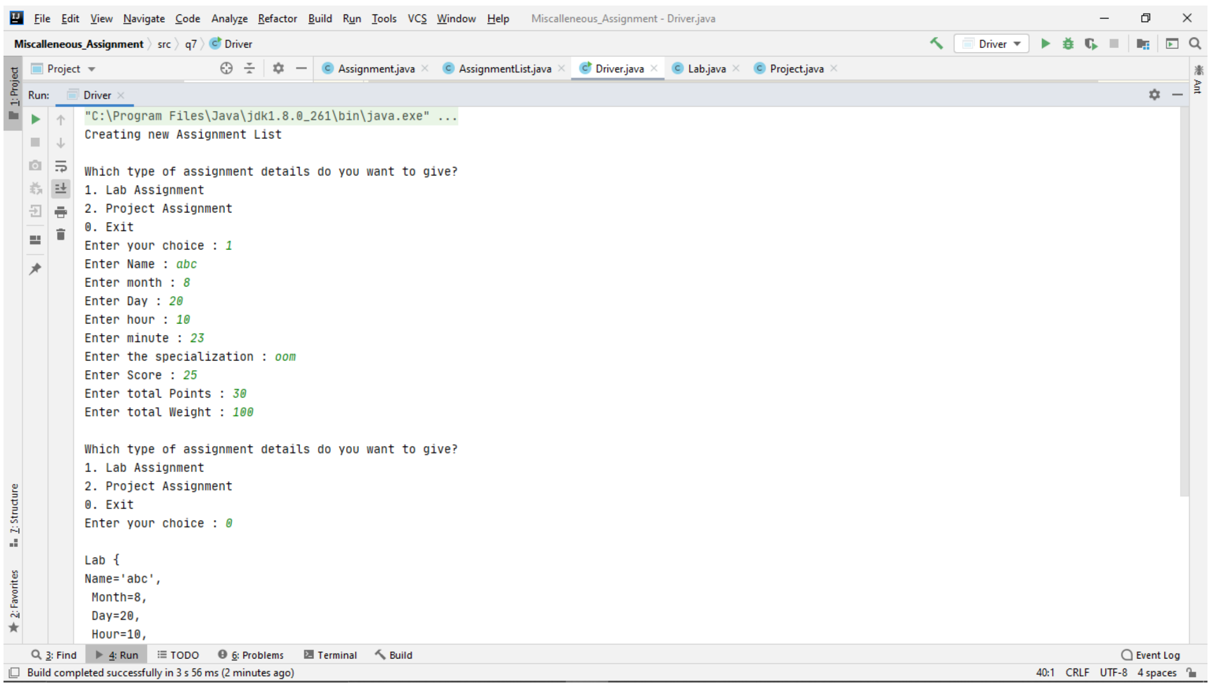
Driver:

```
package q7;
import q7.*;

import java.util.Scanner;

public class Driver {
    public static void main(String[] args) {
        AssignmentList items = new AssignmentList();
        Scanner sc = new Scanner(System.in);
        String name, specialization, dataFile;
        int month, day, hour, minute, totalItems=0;
        boolean var=true;
        while(var){
            System.out.println("\nWhich type of assignment details do you
want to give?");
            System.out.println("1. Lab Assignment");
            System.out.println("2. Project Assignment");
            System.out.println("0. Exit");
            System.out.printf("Enter your choice : ");
            int choice = sc.nextInt();
            switch (choice){
                case 1:
                case 2:     totalItems++;
                            System.out.print("Enter Name : ");
                            name = sc.next();
                            System.out.print("Enter month : ");
                            month = sc.nextInt();
                            System.out.print("Enter Day : ");
                            day = sc.nextInt();
                            System.out.print("Enter hour : ");
                            hour = sc.nextInt();
                            System.out.print("Enter minute : ");
                            minute = sc.nextInt();
                            System.out.print("Enter the specialization :
");
                            specialization = sc.next();
                            if(choice==1) {
                                Assignment lab = new Lab(name, month, day,
hour, minute, specialization);
                                scanSetItems(lab);
                                items.addItem(lab);
                                break;
                            }
                            else{
                                System.out.print("Enter Data File : ");
                                dataFile = sc.next();
                                Assignment project = new Project(name,
month, day, hour, minute, specialization, dataFile);
                                scanSetItems(project);
                                items.addItem(project);
                                break;
                            }
                case 0:     var=false;
                            break;
                default:    System.out.println("Invalid Choice!!! TRY
AGAIN!!!");
            }
        }
        for(int i=0;i<totalItems;++i){
            System.out.println(items.listItems.get(i).toString());
        }

        double grade = items.computeCourseGrade();
        System.out.println("\nCourse Grade : "+ grade);

    }
    static void scanSetItems(Assignment assignment){
        Scanner sc1 = new Scanner(System.in);
        System.out.print("Enter Score : ");
        assignment.setScore(sc1.nextDouble());
        System.out.print("Enter total Points : ");
        assignment.setTotalPoints(sc1.nextDouble());
        System.out.print("Enter total Weight : ");
        assignment.setTotalWeight(sc1.nextDouble());
    }

}
```

Sample Output:

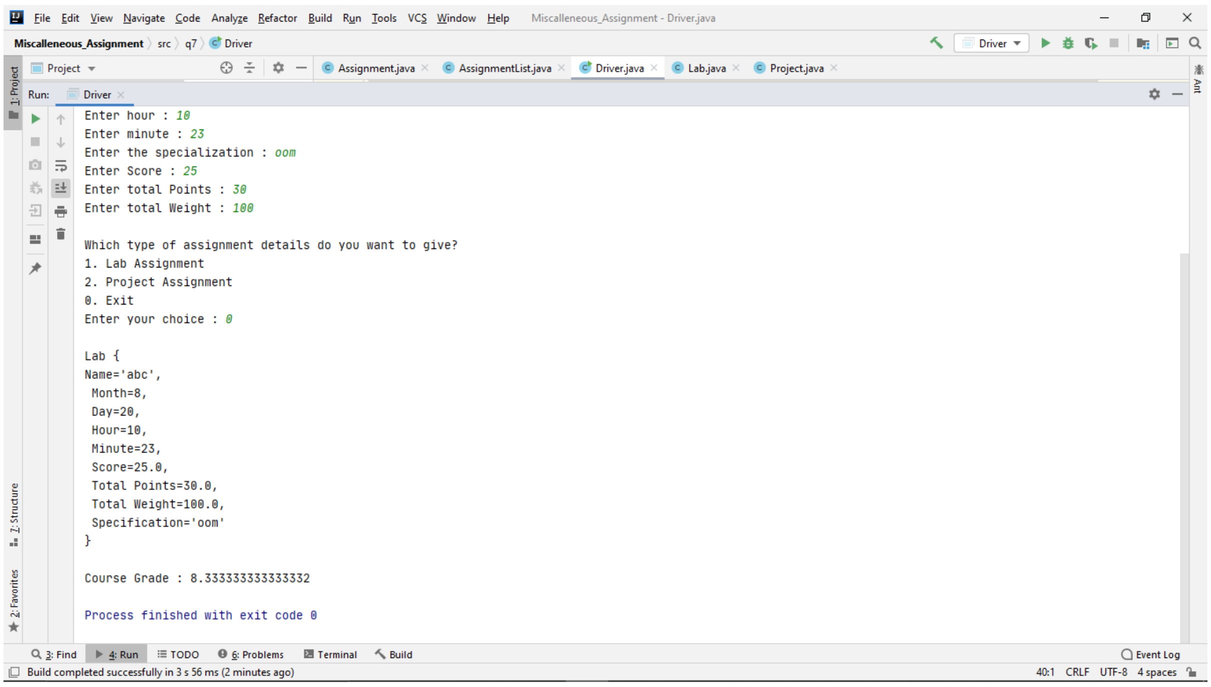
```
Creating new Assignment List

Which type of assignment details do you want to give?
1. Lab Assignment
2. Project Assignment
0. Exit
Enter your choice : 1
Enter Name : abc
Enter month : 8
Enter Day : 20
Enter hour : 10
Enter minute : 23
Enter the specialization : oom
Enter Score : 25
Enter total Points : 30
Enter total Weight : 100

Which type of assignment details do you want to give?
1. Lab Assignment
2. Project Assignment
0. Exit
Enter your choice : 0

Lab {
Name='abc',
Month=8,
Day=20,
Hour=10,
Minute=23,
Score=25.0,
Total Points=30.0,
Total Weight=100.0,
Specification='oom'
}

Course Grade : 8.333333333333332
```

Sample Screenshot:



```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Creating new Assignment List

Which type of assignment details do you want to give?
1. Lab Assignment
2. Project Assignment
0. Exit
Enter your choice : 1
Enter Name : abc
Enter month : 8
Enter Day : 20
Enter hour : 10
Enter minute : 23
Enter the specialization : oom
Enter Score : 25
Enter total Points : 30
Enter total Weight : 100

Which type of assignment details do you want to give?
1. Lab Assignment
2. Project Assignment
0. Exit
Enter your choice : 0

Lab {
Name='abc',
 Month=8,
 Day=20,
 Hour=10,
```

Build completed successfully in 3 s 56 ms (2 minutes ago)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Miscalleneous_Assignment - Driver.java
Miscalleneous_Assignment src q7 Driver
Project
Assignment.java AssignmentList.java Driver.java Lab.java Project.java
Run: Driver
3: Find 4: Run TODO 6: Problems Terminal Build Event Log
Build completed successfully in 3 s 56 ms (2 minutes ago)
40:1 CRLF UTF-8 4 spaces

```
Enter hour : 10
Enter minute : 23
Enter the specialization : oom
Enter Score : 25
Enter total Points : 30
Enter total Weight : 100

Which type of assignment details do you want to give?
1. Lab Assignment
2. Project Assignment
0. Exit
Enter your choice : 0

Lab {
Name='abc',
 Month=8,
 Day=20,
 Hour=10,
 Minute=23,
 Score=25.0,
 Total Points=30.0,
 Total Weight=100.0,
 Specification='oom'
}

Course Grade : 8.333333333333332

Process finished with exit code 0
```