Lab Assignment 06

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Topic:	**Encapsulation, Static variable and Static Method**
Number of Tasks:	**13 (Classwork: 06, Homework: 07)**

[Submit all the Coding Tasks (Homework: Task 1 to 5) in the Google Form shared on buX before the next lab. Submit the Tracing Tasks (Homework: Task 6 to 7) handwritten to your Lab Instructors at the beginning of the lab]

[You are not allowed to change the driver codes of any of the tasks]

CLASSWORK

Task 1

Design the **Account** class such that it produces the following output.

- Each Account will have a name, address and a balance variable.
- The balance must be kept private.
- Use Encapsulation to access and modify balance.

After finishing the class design to get the following output, try to directly print the balance variable. Can you print the private balance variable in the tester code?

Tester Code

```java
public class AccountTester {
  public static void main(String[] args) {
    System.out.print("All accounts belong to ");
    System.out.println(Account.bankName);
    System.out.println("Accounts created: " + Account.count);
    System.out.print("Total Money stored ");
    System.out.println(Account.totalBalance + " taka");
    System.out.println("1--------------------------------");
    Account a1 = new Account("Abir", "Dhanmondi");
    System.out.println("2--------------------------------");
    a1.printInfo();
    System.out.println("3--------------------------------");
    Account a2 = new Account("Mira", "Mirpur");
    a2.printInfo();
    System.out.println("4--------------------------------");
    a1.setBalance(1000);
    System.out.println("5--------------------------------");
    a1.printInfo();
    a2.setBalance(2000);
    System.out.println("6--------------------------------");
    System.out.println(a2.getBalance());
    System.out.println("7--------------------------------");
    a2.setBalance(500);
    System.out.println("8--------------------------------");
    a2.printInfo();
    System.out.println("9--------------------------------");
    System.out.print("All accounts belong to ");
    System.out.println(Account.bankName);
    System.out.println("Accounts created: " + Account.count);
    System.out.print("Total Money stored ");
    System.out.println(Account.totalBalance + " taka");
  }
}
```

Output

```
All accounts belong to Badda Bank
Accounts created: 0
Total Money stored 0.0 taka
1--------------------------------
2--------------------------------
Name: Abir, address: Dhanmondi
Balance: 0.0 Taka
3--------------------------------
Name: Mira, address: Mirpur
Balance: 0.0 Taka
4--------------------------------
5--------------------------------
Name: Abir, address: Dhanmondi
Balance: 1000.0 Taka
6--------------------------------
2000.0
7--------------------------------
8--------------------------------
Name: Mira, address: Mirpur
Balance: 2500.0 Taka
9--------------------------------
All accounts belong to Badda Bank
Accounts created: 2
Total Money stored 3500.0 taka
```

Task 2

Design the **Student** class such that it produces the following output.

Note: A Student needs at least 50 marks to pass.

Tester Code

```java
public class MarksTester {
  public static void main(String[] args) {
    System.out.println("Total Students: " + Student.total_students);
    System.out.println("Average Marks: " + Student.averageMarks());
    System.out.println("----------1----------");
    Student mike = new Student("Mike");
    mike.setCodingMarks(35);
    mike.setTracingMarks(10);
    mike.individualDetail();
    System.out.println("----------2----------");
    System.out.println("Total Students: " + Student.total_students);
    System.out.println("Average Marks: " + Student.averageMarks());
    System.out.println("----------3----------");
    Student eleven = new Student("Eleven", 70, 20);
    eleven.individualDetail();
    System.out.println("----------4----------");
    Student will = new Student("Will");
    will.setCodingMarks(51);
    will.individualDetail();
    System.out.println("----------5----------");
    System.out.println("Total Students: " + Student.total_students);
    System.out.println("Average Marks: " + Student.averageMarks());
  }
}
```

Output

```
Total Students: 0
Average Marks: 0.0
----------1----------
Name: Mike
ID: 1
Coding Marks: 35
Tracing Marks: 10
Mike has failed with 45 marks
----------2----------
Total Students: 1
Average Marks: 45.0
----------3----------
Name: Eleven
ID: 2
Coding Marks: 70
Tracing Marks: 20
Eleven has passed with 90 marks
----------4----------
Name: Will
ID: 3
Coding Marks: 51
Tracing Marks: 0
Will has passed with 51 marks
----------5----------
Total Students: 3
Average Marks: 62.0
```

Task 3

From the given Tester Code write the **Trainer** class that uses the Pokemon class to produce the given output.

- YOU CANNOT change the given Pokemon class.
- Each Trainer can catch at max 7 pokemons.

Tester Code

```java
public class PokemonTrainerTester {
  public static void main(String[] args) {
    Pokemon pikachu = new Pokemon("Pikachu", 65.0);
    Pokemon caterpie = new Pokemon("Caterpie", 45.0);
    Pokemon squirtle = new Pokemon("Squirtle", 70.0);
    Pokemon eevee = new Pokemon("Eevee", 60.0);
    System.out.println("1---------------------");
    Trainer ash = new Trainer("Ash");
    System.out.println("2---------------------");
    ash.catchPokemon(pikachu);
    System.out.println("3---------------------");
    ash.catchPokemon(caterpie);
    ash.catchPokemon(squirtle);
    System.out.println("4---------------------");
    ash.viewPokeDex();
    System.out.println("5---------------------");
    Trainer gary = new Trainer("Gary");
    System.out.println("6---------------------");
    gary.catchPokemon(squirtle);
    gary.catchPokemon(eevee);
    System.out.println("7---------------------");
    gary.viewPokeDex();
    System.out.println("8---------------------");
    Trainer.battle(ash, gary);
  }
}
```

Given Class

```java
public class Pokemon {
  private String name;
  private double hp;
  public Pokemon(String n, double hp) {
    this.name = n;
    this.hp = hp;
  }
  public String getName() {
    return name;
  }
  public double getHP() {
    return hp;
  }
}
```

Output

```
1---------------------
Trainer ID: 1, Name: Ash - created
2---------------------
Ash caught: Pikachu
3---------------------
Ash caught: Caterpie
Ash caught: Squirtle
4---------------------
Ash's Trainer code: T1
Pikachu: 65.0 points
Caterpie: 45.0 points
Squirtle: 70.0 points
Average of HP: 60.0
5---------------------
Trainer ID: 2, Name: Gary - created
6---------------------
Gary caught: Squirtle
Gary caught: Eevee
7---------------------
Gary's Trainer code: T2
Squirtle: 70.0 points
Eevee: 60.0 points
Average of HP: 65.0
8---------------------
Battle Info
Ash's HP average: 60.0
Gary's HP average: 65.0
Gary wins
```

Task 4

Suppose you have opened a new library, from where your friends can borrow books. Initially you have bought 3 books (Pather Panchali, Durgesh Nandini & Anandmath) each of 3 copies only. Design the **Borrower** class in such a way that the following code provides the expected output.

- You are given the arrays ***book_count*** and ***book_name*** to keep track of the number of books available. For simplicity, assume that there will be no other books in the library.
- You must reuse the ***remainingBooks()*** method when needed.

Given Code

```java
public class BorrowerTester{
  public static void main(String args[]){
    Borrower.bookStatus();
    System.out.println("*********1*********");
    Borrower b1 = new Borrower("Nabila");
    b1.borrowBook("Pather Panchali");
    b1.borrowBook("Anandmath");
    b1.borrowerDetails();
    System.out.println("*********2*********");
    Borrower b2 = new Borrower("Sadia");
    b2.borrowBook("Anandmath");
    b2.borrowBook("Durgesh Nandini");
    b2.borrowBook("Pather Panchali");
    b2.borrowerDetails();
    System.out.println("*********3*********");
  System.out.println(Borrower.remainingBooks("Anandmath")+"
copies of Anandmath is remaining.");
    System.out.println("*********4*********");
    Borrower b3 = new Borrower("Anika");
    b3.borrowBook("Anandmath");
    Borrower.bookStatus();
    System.out.println("*********5*********");
    Borrower b4 = new Borrower("Oishi");
    b4.borrowBook("Anandmath");
    b4.borrowBook("Durgesh Nandini");
    b4.borrowerDetails();
  }
}

public class Borrower{
  public static int book_count[] = {3, 3, 3};
  public static String book_name[] = {"Pather Panchali",
"Durgesh Nandini", "Anandmath"};

  // Your Code here
}
```

Expected Output

```
Available Books:
Pather Panchali: 3
Durgesh Nandini: 3
Anandmath: 3
*********1*********
Name: Nabila
Books Borrowed:
Pather Panchali
Anandmath
*********2*********
Name: Sadia
Books Borrowed:
Anandmath
Durgesh Nandini
Pather Panchali
*********3*********
1 copies of Anandmath is
remaining.
*********4*********
Available Books:
Pather Panchali: 1
Durgesh Nandini: 2
Anandmath: 0
*********5*********
This book is not available.
Name: Oishi
Books Borrowed:
Durgesh Nandini
```

Task 5

```java
public class B{
  public static int x;
  public int y = 4;
  public int temp = -5;
  public int sum = 2;
  public B(){
    y = temp + 3 ;
    sum = 3 + temp + 3;
    temp-=2;
  }
  public B(B b){
    sum = b.sum;
    x = b.x;
    b.methodB(1,3);
  }
   public void methodA(int m, int n){
    int x = 2;
    y = y + m + (temp++);
    x = x + 7 +  n;
    sum = sum + x + y;
    System.out.println(x + " " + y+ " " + sum);
  }
  public void methodB(int m, int n){
    int  y = 0;
    y = y + this.y;
    x = this.y + 3 + temp;
    methodA(x, y);
    sum = x + y + sum;
    System.out.println(x + " " + y+ " " + sum);
  }
}
```

Consider the following driver code and find the output.

```java
B b1 = new B();
B b2 = new B(b1);
b1.methodA(3, 2);
b2.methodB(1, 2);
```

x	y	sum

Task 6

```
class Trace6CW {
 public static int temp = 10;
 public int sum = 0;
 public static int[] y = {2, 5};
 public Trace6CW() {
   temp -= 2;
   sum = temp + y[0];
   y[1] = sum - temp;
 }
 public void methodA(int m, int n) {
   int temp = 0;
   temp = Trace6CW.temp + m;
   this.sum = this.sum + temp + Trace6CW.y[1];
   Trace6CW.y[0] = this.sum - n;
   System.out.println(this.sum + " " + temp + " " + Trace6CW.y[0]);
 }
 public static void methodB(Trace6CW s1, int m) {
   int sum = 5;
   y[0] = s1.sum + sum;
   s1.sum = temp + y[1] + m;
   System.out.println(s1.sum + " " + y[0] + " " + temp);
 }
}
```

```
class Test5CW {
  public static void main(String[] args) {
    Trace6CW s1 = new Trace6CW();
    Trace6CW s2 = new Trace6CW();
    s1.methodA(3, 2);
    Trace6CW.methodB(s2, 4);
    s2.methodA(1, 1);
  }
}
```

OUTPUT		

HOMEWORK

Task 1

Design the **Product** class such that it produces the following output.

Tester Code

```
public class ProductTester{
  public static void main(String[] args) {
    Product p1 = new Product("Table", 10);
    Product p2 = new Product("Chair", 15);
    Product p3 = new Product("Sofa", 20);
    Product p4 = new Product("Divan", 8);
    System.out.println("------------1------------");
    Product.displayProducts();
    System.out.println("------------2------------");
    Product.buy("Chair", 5);
    System.out.println("------------3------------");
    Product.displayProducts();
    System.out.println("------------4------------");
    Product.buy("Sofa", 25);
    System.out.println("------------5------------");
    Product.displayProducts();
    System.out.println("------------6------------");
    Product.buy("Bed", 10);
  }
}
```

Output

```
Stored: Table
Stored: Chair
Stored: Sofa
Storage is full! Cannot add Divan
------------1------------
=== Stored Products ===
Table - Qty: 10
Chair - Qty: 15
Sofa - Qty: 20
------------2------------
Product Sold
------------3------------
=== Stored Products ===
Table - Qty: 10
Chair - Qty: 10
Sofa - Qty: 20
------------4------------
Quantity low
------------5------------
=== Stored Products ===
Table - Qty: 10
Chair - Qty: 10
Sofa - Qty: 20
------------6------------
Product not found
```

Task 2

Design the **Character** class such that it produces the following output.

Tester Code

```java
public class WeirderStuffTester {
  public static void main(String[] args) {
    Character.printStats();
    System.out.println("---------1-----------");
    Character twelve = new Character("Twelve", "Kid", 100);
    twelve.printDetails();
    System.out.println("---------2-----------");
    Character spike = new Character("Spike", "Kid", 50);
    spike.printDetails();
    System.out.println("---------3-----------");
    Character.printStats();
    System.out.println("---------4-----------");
    Character reeve = new Character("Reeve", 70);
    reeve.printDetails();
    System.out.println("---------5-----------");
    Character chopper = new Character("Chopper", "Adult", 120);
    chopper.printDetails();
    System.out.println("---------6-----------");
    Character.printStats();
  }
}
```

Output

```
Total Characters: 0
Kids: 0
Teens: 0
Adults: 0
Average Health: 0
Strongest Character: None
---------1-----------
ID: 1, Name: Twelve
Group: Kid
Health: 100
---------2-----------
ID: 2, Name: Spike
Group: Kid
Health: 50
---------3-----------
Total Characters: 2
Kids: 2
Teens: 0
Adults: 0
Average Health: 75.0
Strongest Character:
Twelve (Health 100)
---------4-----------
ID: 3, Name: Reeve
Group: Teen
Health: 70
---------5-----------
ID: 4, Name: Chopper
Group: Adult
Health: 120
---------6-----------
Total Characters: 4
Kids: 2
Teens: 1
Adults: 1
Average Health: 85.0
Strongest Character:
Chopper (Health 120)
```

Task 3

Design the **Artifact** class where all the attributes of the class are classified i.e. private. The "Vault" can only store a maximum of 4 artifacts. The power of the artifacts are calculated as: For artifact with,

- Even length name => Summation of all characters in even index of name.
- Odd length name => Summation of all characters in odd index of name.

Tester Code

```
public class TesterArtifact{
  public static void main(String[] args) {
    Artifact a = new Artifact("Phone Microwave",
"Kurisu");
    System.out.println("--------1-------");
    Artifact.AddtoVault(a);
    Artifact.AddtoVault(new Artifact("D-Mail Capsule",
"Mayuri"));
    System.out.println("--------2-------");
    Artifact c = new Artifact("C204 Chip");
    Artifact d = new Artifact("Divergence Meter");
    Artifact e = new Artifact("M4A2 Robot", "Okabe");
    Artifact.AddtoVault(c);
    Artifact.AddtoVault(d);
    Artifact.AddtoVault(e);
    System.out.println("--------3-------");
    Artifact.labReport();
    System.out.println("--------4-------");
    System.out.println("Power of "+c.GetName()+" is
"+c.CalcPower());
    System.out.println("--------5-------");
    System.out.println("Strongest Artifact:
"+Artifact.strongest());
    System.out.println("--------6-------");
    a.revealArtifact();
    System.out.println("--------7-------");
    a.changeName("Banana Microwave");
    System.out.println("--------8-------");
    Artifact.labReport();
    System.out.println("--------9-------");
    System.out.println("Strongest Artifact:
"+Artifact.strongest());
  }
}
```

Output

```
--------1-------
Kurisu added Phone Microwave successfully to the
vault.
Mayuri added D-Mail Capsule successfully to the
vault.
--------2-------
Okabe added C204 Chip successfully to the vault.
Okabe added Divergence Meter successfully to the
vault.
!!Okabe unsuccessful in adding artifact to the
vault!!
--------3-------
=== Future Gadget Lab ===
Phone Microwave added by Kurisu has power of
702.
D-Mail Capsule added by Mayuri has power of 602.
C204 Chip added by Okabe has power of 274.
Divergence Meter added by Okabe has power of
734.
--------4-------
Power of C204 Chip is 274
--------5-------
Strongest Artifact: Divergence Meter
--------6-------
Phone Microwave added by Kurisu has power of
702.
--------7-------
Name changed and power recalculated.
--------8-------
=== Future Gadget Lab ===
Banana Microwave added by Kurisu has power of
774.
D-Mail Capsule added by Mayuri has power of 602.
C204 Chip added by Okabe has power of 274.
Divergence Meter added by Okabe has power of
734.
--------9-------
Strongest Artifact: Banana Microwave
```

Task 4

Design the **AnimalKeepers** class with the following requirements:

- The Animal Keepers have private IDs starting from 101.
- The Safari has an array named, Animals = {“Lion”, “Tiger”, “Seal”, “Gorilla”, “Deer”}
- Only one task is assigned per animal and so the tasks are overridden when reassigned for the same animal.

[Hint: You can call the static method printTasks() from inside details()]

Tester Code

```java
import java.util.Arrays;
public class KeeperTester {
  public static void main(String[] args) {
    Animalkeepers.details();
    System.out.println("---------1----------");
    System.out.println(Arrays.toString(Animalkeepers.Animals));
    System.out.println("---------2----------");
    Animalkeepers leo = new Animalkeepers("Leo");
    Animalkeepers theo = new Animalkeepers("Theo");
    Animalkeepers mochi = new Animalkeepers("Mochi");
    System.out.println("---------3----------");
    Animalkeepers.printTasks();
    System.out.println("---------4----------");
    leo.doTask("Lion", "Feed");
    System.out.println("---------5----------");
    leo.doTask("Monkey", "Feed");
    System.out.println("---------6----------");
    Animalkeepers.details();
    System.out.println("---------7----------");
    theo.doTask("Tiger", "Bathe");
    mochi.doTask("Seal", "Clean Pen");
    mochi.doTask("Deer", "Add Food");
    System.out.println("---------8----------");
    Animalkeepers.printTasks();
    System.out.println("---------9----------");
    leo.doTask("Deer", "Play");
    System.out.println("---------10----------");
    Animalkeepers.details();
  }
}
```

Output

```
No Animal Keepers working yet.
---------1----------
[Lion, Tiger, Seal, Gorilla, Deer]
---------2----------
Leo with ID 101 got the job!
Theo with ID 102 got the job!
Mochi with ID 103 got the job!
---------3----------
No tasks assigned.
---------4----------
Task assigned to Leo
---------5----------
Animal not in the Safari
---------6----------
Total Animal Keeper: 3
Total Task assigned: 1
Feed (Keeper - Leo) === Lion
---------7----------
Task assigned to Theo
Task assigned to Mochi
Task assigned to Mochi
---------8----------
Feed (Keeper - Leo) === Lion
Bathe (Keeper - Theo) === Tiger
Clean Pen (Keeper - Mochi) === Seal
Add Food (Keeper - Mochi) === Deer
---------9----------
Task assigned to Leo
---------10----------
Total Animal Keeper: 3
Total Task assigned: 4
Feed (Keeper - Leo) === Lion
Bathe (Keeper - Theo) === Tiger
Clean Pen (Keeper - Mochi) === Seal
Play (Keeper - Leo) === Deer
```

Task 5

Design the **Event** and **Organizer** classes in such a way that the following code provides the expected output. Hint:

- Make the name instance variable of the Event class **private**
- For simplicity assume that the Event class can create a maximum of 5 event objects and an Organizer can organize a maximum of 4 events.

Driver Code

```java
public class EventTester{
  public static void main(String args []){
    Event.allEventInfo();
    System.out.println("1--------------");
    Event ev1 = new Event("HP Day", "7/12/24");
    Event ev2 = new Event("TechConnect", "10/12/24");
    System.out.println(ev1.details());
    System.out.println("2--------------");
    Organizer uni = new Organizer();
    Organizer bracu = new Organizer("BRACU");
    Organizer buet = new Organizer("BUET");
    System.out.println("3--------------");
    Event.allEventInfo();
    System.out.println("4--------------");
    bracu.organizeEvent(ev1);
    bracu.organizeEvent(ev2);
    System.out.println("5--------------");
    Event ev3 = new Event("From Earth to Orbit",
"15/12/24");
    Event ev4 = new Event("NSysS 2024","21/12/24");
    System.out.println("6--------------");
    buet.organizeEvent(ev4);
    bracu.organizeEvent(ev3);
    System.out.println("7--------------");
    bracu.searchEventByDate("21/12/24");
    System.out.println("8--------------");
    bracu.searchEventByDate("15/12/24");
    System.out.println("9--------------");
    Event.allEventInfo();
  }
}
```

Output

```
Total Events: 0
Event Details:
1--------------
Name: HP Day
Date: 7/12/24
2--------------
Please provide the organizer's name
3--------------
Total Events: 2
Event Details:
Event 1:
Name: HP Day
Date: 7/12/24
Event 2:
Name: TechConnect
Date: 10/12/24
4--------------
BRACU successfully organized HP Day
BRACU successfully organized TechConnect
5--------------
6--------------
BUET successfully organized NSysS 2024
BRACU successfully organized From Earth to
Orbit
7--------------
No event is scheduled for 21/12/24
8--------------
From Earth to Orbit
9--------------
Total Events: 4
Event Details:
Event 1:
Name: HP Day
Date: 7/12/24
Event 2:
Name: TechConnect
Date: 10/12/24
Event 3:
Name: From Earth to Orbit
Date: 15/12/24
Event 4:
Name: NSysS 2024
Date: 21/12/24
```

Task 6

```java
class Trace {
 public static int[] x = {3, -4};
 public int y = 4;
 public static int temp = -5;
 private int sum = 2;
 public Trace(){
   y = temp + 3 ;
   sum = 3 + temp + x[1];
   temp-=2;
   x[0] = ++x[1] - 2;
 }
 public Trace(Trace trace){
   sum = trace.sum;
   x = trace.x;
   trace.methodB(1,3);
 }
 public void methodA(int m, int n){
   int x = 2 - this.x[0] - Trace.x[1];
   y = y + m + (temp++);
   x = x + 7 +  n;
   sum = sum + x + y;
   System.out.println(x + " " + y+ " " + sum);
 }
 public void methodB(int m, int n){
   int  y = 0;
   y = y + this.y;
   Trace.x[0] = this.y + 3 + temp;
   methodA(x[1], y);
   sum = Trace.x[1] + y + sum;
   System.out.println(this.x[0] + " " + y+ " " + sum);
 }
 public static void methodC(Trace trace1, Trace trace2){
   temp = x[0] - Trace.x[1];
   x = new int[]{trace1.y, trace2.y};
 }
}
```

Consider the following driver code and find the output.

```java
Trace trace1 = new Trace();
Trace trace2 = new Trace(trace1);
trace1.methodA(3, 2);
Trace.methodC(trace1, trace2);
trace2.methodB(1, 2);
```

Output 1	Output 2	Output 3

Task 7

```java
class Tracing {
 public static int x = 0, y = 0;
 public int a;
 private int b = 3;
 public Tracing(int a, int b) {
   this.a = a;
   this.b = b - this.b;
   x += 1;
   y += Tracing.y - 2;
 }
 public void set_b(int b) {
   this.b = b;
 }
 public int get_b() {
   return this.b;
 }
 public void methodA(int x) {
   this.a = x + this.x - Tracing.x;
   this.b = this.a + this.methodB() - this.b;
   System.out.println(this.a + " " + this.b + " " + x);
 }
 public int methodB() {
   int y = -3;
   this.b = y - this.y + this.a;
   System.out.println(this.a + " " + this.b + " " + x);
   this.y -= y;
   x += this.b + this.y;
   return this.b;
 }
 public void methodB(Tracing t1) {
   int t = this.y - t1.get_b() + this.b;
   t1.set_b(t);
   t1.a = this.x - t1.a + this.a;
   System.out.println(t1.a + " " + t1.get_b() + " " + x);
 }
}
```

Consider the following driver code and find the output.

```java
Tracing t1 = new Tracing(2, 3);
t1.methodA(1);
Tracing t2 = new Tracing(3, 4);
t2.methodA(2);
t1.methodB(t2);
t2.methodB(t2);
```

Output 1	Output 2	Output 3

Ungraded Tasks (Optional)

(You don’t have to submit the ungraded tasks)

Task 1

Design the **SultansDine** class with the necessary property to produce the output from the given driver code.

Subtaks:

1. Create SultansDine class
2. Create 2 static variable and 1 static array
3. Create 1 static method
4. Calculation of branch sell is given below
    a. If sellQuantity < 10:
        i. Branch_sell = quantity * 300
    b. Else if sellQuantity < 20:
        i. Branch_sell = quantity * 350
    c. Else
        i. Branch_sell = quantity * 400
5. Calculation of branch’s sell percentage = (branch’s sell / total sell) * 100

Driver Code

```java
public class SultansDineTester {
  public static void main(String[] args) {
    SultansDine.details();
    System.out.println("1===================");
    SultansDine dhanmondi = new SultansDine("Dhanmondi");
    dhanmondi.sellQuantity(25);
    dhanmondi.branchInformation();
    System.out.println("2===================");
    SultansDine.details();
    System.out.println("3===================");
    SultansDine baily_road = new SultansDine("Baily
Road");
    baily_road.sellQuantity(15);
    baily_road.branchInformation();
    System.out.println("4===================");
    SultansDine.details();
    System.out.println("5===================");
    SultansDine gulshan = new SultansDine("Gulshan");
    gulshan.sellQuantity(9);
    gulshan.branchInformation();
    System.out.println("6===================");
    SultansDine.details();
  }
}
```

Output

```
Total Number of branch(s): 0
Total Sell: 0 Taka
1===================
Branch Name: Dhanmondi
Branch Sell: 10000 Taka
2===================
Total Number of branch(s): 1
Total Sell: 10000 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total sell's 100.00
3===================
Branch Name: Baily Road
Branch Sell: 5250 Taka
4===================
Total Number of branch(s): 2
Total Sell: 15250 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total sell's 65.57
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total sell's 34.43
5===================
Branch Name: Gulshan
Branch Sell: 2700 Taka
6===================
Total Number of branch(s): 3
Total Sell: 17950 Taka
Branch Name: Dhanmondi, Branch Sell: 10000 Taka
Branch consists of total sell's 55.71
Branch Name: Baily Road, Branch Sell: 5250 Taka
Branch consists of total sell's 29.25
Branch Name: Gulshan, Branch Sell: 2700 Taka
Branch consists of total sell's 15.04
```

Task 2

Design the required class/es so that the following output is generated. Read the following description:

1. You may assume that to board a bus, a student must have the bus pass, and his/her location must match the route of the bus.
2. Additionally, the default maximum capacity of the bus is 2.
3. The location attribute of the Student class will be **private**

Driver Code

```java
public class BracuStudentTester {
 public static void main(String[] args) {
  BracuStudent st1 = new BracuStudent("Afif", "Mirpur");
  System.out.println("1===============");
  BracuStudent st2 = new BracuStudent("Shanto", "Motijheel");
  BracuStudent st3 = new BracuStudent("Taskin", "Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("2===============");
  st3.showDetails();
  System.out.println("3===============");
  BracuBus bus1 = new BracuBus("Mirpur");
  BracuBus bus2 = new BracuBus("Azimpur", 5);
  bus1.showDetails();
  bus2.showDetails();
  System.out.println("4===============");
  st2.collectPass();
  st3.collectPass();
  System.out.println("5===============");
  st2.showDetails();
  st3.showDetails();
  System.out.println("6===============");
  bus1.board();
  System.out.println("7===============");
  bus1.board(st1, st2);
  System.out.println("8===============");
  st1.collectPass();
  st2.setLocation("Mirpur");
  st1.showDetails();
  st2.showDetails();
  System.out.println("9===============");
  bus1.board(st1);
  bus1.board(st2, st3);
  System.out.println("10===============");
  bus1.showDetails();
 }
}
```

Output

```
1===============
Student Name: Afif
Lives in Mirpur
Have Bus Pass? false
Student Name: Shanto
Lives in Motijheel
Have Bus Pass? false
2===============
Student Name: Taskin
Lives in Mirpur
Have Bus Pass? false
3===============
Bus Route: Mirpur
Passenger Count: 0 (Max: 2)
Passengers on Board:
Bus Route: Azimpur
Passenger Count: 0 (Max: 5)
Passengers on Board:
4===============
5===============
Student Name: Shanto
Lives in Motijheel
Have Bus Pass? true
Student Name: Taskin
Lives in Mirpur
Have Bus Pass? true
6===============
No passengers
7===============
You don't have a bus pass!
You got on the wrong bus!
8===============
Student Name: Afif
Lives in Mirpur
Have Bus Pass? true
Student Name: Shanto
Lives in Mirpur
Have Bus Pass? true
9===============
Afif boarded the bus.
Shanto boarded the bus.
Bus is full!
```

	```
	10===============
	Bus Route: Mirpur
	Passenger Count: 2 (Max: 2)
	Passengers on Board:
	Afif Shanto
	```

Task 3

Design a **Student** class in such a way that the following code provides the expected output.

Driver Code	Output

```
public class StudentTester {
  public static void main(String[] args) {
    Student.printDetails();
    System.out.println("--------------------");
    Student mikasa = new Student("Mikasa", 3.75);
    mikasa.individualDetail();
    System.out.println("--------------------");
    Student.printDetails();
    System.out.println("--------------------");
    Student harry = new Student("Harry", 2.5, "Charms");
    harry.individualDetail();
    System.out.println("--------------------");
    Student.printDetails();
    System.out.println("--------------------");
    Student levi = Student.createStudent("Levi", 3.33);
    levi.individualDetail();
    System.out.println("--------------------");
    Student.printDetails();
  }
}
```
	```
	Total Student(s): 0
	CSE Student(s): 0
	Other Department Student(s): 0
	--------------------
	ID: 1
	Name: Mikasa
	CGPA: 3.75
	Department: CSE
	--------------------
	Total Student(s): 1
	CSE Student(s): 1
	Other Department Student(s): 0
	--------------------
	ID: 2
	Name: Harry
	CGPA: 2.5
	Department: Charms
	--------------------
	Total Student(s): 2
	CSE Student(s): 1
	Other Department Student(s): 1
	--------------------
	ID: 3
	Name: Levi
	CGPA: 3.33
	Department: CSE
	--------------------
	Total Student(s): 3
	CSE Student(s): 2
	Other Department Student(s): 1
	```

Task 4

For this task, you need to design the **Cargo** class with appropriate static and non-static variables and methods to produce this given output for the given tester code.
Note: .load() method marks an object as selected for transport, and .unload() method unmarked it. At a time, the transport capacity is 10.0 tonnes. Each Cargo object is initialized with 2 attributes from the constructor - the contents and the weight. Carefully observe the outputs to identify the other attributes and design the class.

Given Code

```java
public class CargoTester {
  public static void main(String[] args) {
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
    System.out.println("1====================");
    Cargo a = new Cargo("Industrial Machinery", 4.5);
    a.details();
    System.out.println("2====================");
    a.load();
    System.out.println("3====================");
    Cargo b = new Cargo("Steel Ingot", 2.7);
    b.details();
    System.out.println("4====================");
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
    System.out.println("5====================");
    b.load();
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
    System.out.println("6====================");
    Cargo c = new Cargo("Tree Trunks", 3.6);
    c.load();
    System.out.println("7====================");
    c.details();
    b.details();
    System.out.println("8====================");
    Cargo d = new Cargo("Processed Goods", 1.8);
    d.load();
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
    System.out.println("9====================");
    b.unload();
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
    System.out.println("10====================");
    c.load();
    System.out.println("11====================");
    b.details();
    System.out.println("Cargo Capacity: "+
Cargo.capacity());
  }
}
```

Expected Output

```
Cargo Capacity: 10.0
1====================
Cargo ID: 1, Contents: Industrial
Machinery, Weight: 4.5, Loaded:
false
2====================
Cargo 1 loaded for transport.
3====================
Cargo ID: 2, Contents: Steel
Ingot,
Weight: 2.7, Loaded: false
4====================
Cargo Capacity: 5.5
5====================
Cargo 2 loaded for transport.
Cargo Capacity: 2.8
6====================
Cannot load cargo, exceeds weight
capacity.
7====================
Cargo ID: 3, Contents: Tree
Trunks,
Weight: 3.6, Loaded: false
Cargo ID: 2, Contents: Steel
Ingot,
Weight: 2.7, Loaded: true
8====================
Cargo 4 loaded for transport.
Cargo Capacity: 1.0
9====================
Cargo 2 unloaded.
Cargo Capacity: 3.7
10====================
Cargo 3 loaded for transport.
11====================
Cargo ID: 2, Contents: Steel
Ingot,
Weight: 2.7, Loaded: false
Cargo Capacity:
0.09999999999999964
```

Task 5

```
public class Maze{
    public static int x;
    public void methodA(){
      int m = 5;
      x=11;
      System.out.println(x+" "+m);
      m=methodB(m-3)+x;
      System.out.println(x+" "+(m));
      methodB(x,m);
      System.out.println(x+" "+m+x);
    }
    public int methodB(int y){
      x=y*y;
      System.out.println(x+" "+y);
      return x+3;
    }
    public void methodB(int z, int x){
      z=z-2;
      x=x*1%z;
      System.out.println(z+" "+x);
    }
}
public class TestU3{
      public static void main(String [] args){
          Maze c = new Maze();
          c.methodA();
          c.methodB(-11, 45);
      }
}
```

Output	

Task 6

```
class TraceTask {
 public static int x = 3;
 public int y = 5;
 public int z = 2;
 public TraceTask(int z) {
   this.z = z;
   x += 2;
   this.y = x - this.z;
 }
 public void methodA(int x) {
   this.y = x + this.y + TraceTask.x;
   x = this.z + 4;
   TraceTask.x = this.y - x;
   System.out.println(x + " " + this.y + " " + TraceTask.x);
 }
 public int methodB(TraceTask t, int z) {
   int y = 2;
   t.z = this.z + y;
   t.methodA(y);
   this.x = t.y + this.z;
   System.out.println(this.z + " " + t.z + " " + y);
   return this.x;
 }
}
```

```
class TraceTaskTester{
    public static void main(String[] args) {
        TraceTask t1 = new TraceTask(4);
        TraceTask t2 = new TraceTask(2);
        t1.methodA(3);
        int res = t2.methodB(t1, 5);
       System.out.println(res + " " + TraceTask.x);
    }
}
```

	OUTPUT	