Name - ARPAN MANDAL

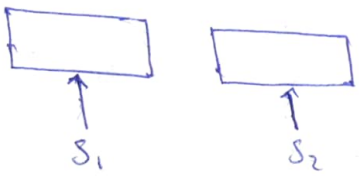
Class roll no. - 001910501061

Exam roll no. - CSE214021 (1)

Part-A

1.

a) S_1 and S_2 are referring to the instances of same class.



(i) $(S_1 == S_2)$:

Here not the instances of the class but reference are compared

(ii) S_1.equals(S_2):

Here we have to override equals methods in the class to compare S_1 & S_2.

Here If we don't define equals methods, then reference will be compared.

b) Before the time of object destruction by garbage collector.

b) Before the time of object destruction by garbage collector, there method finalize() is invoked on behalf of the object beg being removed if it is defined in the class

defenation.

```
protected void finalize(){
        :
        :
}
```

(2)

c)

```
class X {
    :
    :
    f(int)
}
class Y extends X {
    :
    :
    f(float)
}
```

```
class X {
    :
    :
    f(int)
}
class Y extends X {
    :
    :
    f(float)
}
```

```
int i; float fl;
Y e = new Y();   X b = e;
b.f(fl);
```

Here e is the object of derived class Y.

e.f(i) // Here e can able to call f(int) of the X class successfully because e is the object of derived class so it call call the same signature function in the base class

b.f(fl) // This line will generate an error because the base class object b dont able to access the method of derived class f(float) which is overridden function of f(int) of Base class X.

d) Utility of inner class in Java:-

1) Nested classes represent a special type of relationship that is it can access all the members of outer class including private.

2) Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.

Suppose a particular class A is required only for a specific class B. make A nested class in B

```
class B{

 class A{

  }
}
```

So, It provide ~~better~~ encapsulation

e) The finally block is used to put important codes such as clean up, closing the file, closing the connection that must run at the end of the execution

```
try{        → So here execution of try or
}               catch block, finally block
catch{          is executed.
}
catch{ → finally{
;
}
```

(2). 2.

a) There is two way to specify the code for a thread and data on which it work.

1) Extend Thread Class :-

```
Class MyThread extends Thread{
   int data;
   void code(){
      - - -
   }
   MyThread(int x){
      data = x;
   }
   public void run(){
        code();
   }
main(){
   MyThread Mt = new MyThread(data);
   Mt.start();
```

2) Implement Runnable interface :-

```
Class MyThread implements Runnable {
   int data;
   void code(){
      - -
   }
   MyThread(int x){
      data = x;
   }
   Public void run(){
        code();
   }
}
```

```
main() {
MyThread Mt = new MyThread(data);
Thread t= new Thread(mt);
t.start();
}
```

So in both methode we can use a Construcor of MyThread class to specify the data on which it performs operation and to specify which operation it perform over the data & we call this method in the run method as shown above, we can also specify the run method to perform operation on the data.

b) notifyall() wakesup all the threads which are waiting.

notify() wakes up only one thread among all the waiting thread.

e.g.

Message class-

```
Public class Message{
    String msg;

    Message (String s){
        msg = s
    }

}
```

Waiter class.

```
Public class waiter implements Runnable {
    private Message msg;
    waiter (Message m){
        msg = m;
    }
    public void run() {
        Synconised(msg){
        try{
          msg.wait()
        }
        catch(--){
        }
        }
    }
}
```

notifier class

```
Public class Notifier implements Runnable {
    private Message msg,
    Notifier (Message m){
        msg = m
    }
  Public void run(){
      try {
      Synconized (msg){
          msg.notifyAll(); //msg.notify()
      }
      }
      catch(--) {
      }
  }
}
```

Main class

```
Public class Main{
  Public static void main(String args[]){
    Message msg= new Message("process it");
    Waiter waiter= new Waiter (msg);
    new Thread(waiter,"waiter").start();
    Waiter waiter1_second= new Waiter (msg);
    new Thread(waiter1,"wait
    new Thread (waiter_second,"waiter_second")
                                .start();
    Notifier notifier= new notifier(msg);
    new Thread(notifier,"notifier").start();
  }
}
```

here whenever waiter and waiter_second thread is running and go through try block then these two thread will have to wait. whenever notifier thread is running, it will awake both the thread. If we use notify() then we won't know which thread will run but notifyAll() will awake all the Thread. That's why it is preferred.

4.

a) Comparison between panel and frame :-

Panel in Java	Frame in Java
(i) An AWT Component which represents a simple container that can attach other GUI components including other panels	(i) An AWT Component which is a top level window with border and title
(ii) A subclass of Container	(ii) A subclass of window
(iii) public class panel extends Container implements Accessible.	(iii) public class Frame extends window implements MenuContainer
(iv) Doesn't have a title bar or border	(iv) have a title bar and border.

b)

```
import javax.swing.*;
import java.awt.event.*;

class OptionSelector extends JFrame
implements ActionListener{

JRadioButton hBtn1, hBtn2, hBtn3;

JButton button_click;

OptionSelector(){
  hBtn1 = new JRadioButton("option1");
  hBtn1.SetBounds(100, 50, 100, 30);
  hBtn2 = new JRadioButton("option2");
  hBtn2.SetBounds(100, 100, 100, 30);
```

```
rBtn3 = new JRadioButton ("option 3");
rBtn3.SetBounds (100,150,100,30);
ButtonGroup bGroup = new ButtonGroup();
bGroup.add (rBtn1);
bGroup.add (rBtn2);
bGroup.add (rBtn3);
button_click = new JButton ("click");
button_click.SetBounds (100,200,80,30);
button_click.addActionListener (this);
add (rBtn1);
add (rBtn2);
add (rBtn3);
add (button_click);
setSize (300,350);
setLayout (null);
setVisible (true);
SetDefaultCloseOperation(EXIT_ON_CLOSE);
}
Public void actionperformed (ActionEvent e){
    if (rBtn1.isSelected()){
        JOptionpane.showMessageDialog (this,
                                       "option 1 selected");
    }
    if (rBtn2.isselected()){
        JOptionpane.ShowMessageDialog (this,
                                       "option 2 selected");
    }
```

```
        if (bBtn3.isSelected()){
            JOptionPane.showMessageDialog(this, "option-3
                                                selected");
        }
    }
    public static void main(String args[]){
            new OptionSelector();
        }
}
```

6. a)

```
import java.io.*;
Public class STUDENT{
    String name;
    String roll;
    int score;
    STUDENT(String name, String roll, int score){
        this.name = name;
        this.roll = roll;
        this.score = score;
    }
    @override
    public String toString(){
        return ("Name:" + Name + "\t roll:" + roll +
                + "\t score:" + score);
    }
Public class FILEIO{
    Static void WriteToFile(String filename, Objects){
        try{
            FileOutputStream f = new FileOutputStream
                                 (filename,
                                  true);
        ObjectOutputStream obj = new ObjectOutputStream(f);
```

```
        obj.writeObject(s);
        obj.close();
      }
      catch(Exception e){
          e.getStackTrace();
      }
  }
  Static void headFromFile(String filename){
      try{
      FileInputStream f=new FileInputStream
                              ("filename");
      ObjectOutputStream obj=new objectoutput-
                              -Stream(f);
      Object s=null;
      while((s=obj.readObject())!=null){
          System.out.println(s);
      }
      obj.close();
      }
      catch(Exception e){
          e.getStackTrace();
      }
  }
  Public Static void main(String args[]){
      try{
      Student1 s
      STUDENT s=new STUDENT("ABC","0x191055162
                              62);
      writeToFile("output",s);
      headFromFile("output");
      }
      catch(Exception e){e.fillInStackTrace();}
  }
}
```

b) In order to use the Contains() method of the collections we should override the equals methode and also hashCode() methode to implement it for STUDENT Class. Here overriding of hashCode() is not mendetory. e.g. -

```
Class STUDENT{
  String Name;
  String roll;
  int score;

  @override
  public boolean equals (String roll){
    if (this.roll).equals(roll){
      return true;
    }
    return ((this.roll).equals(roll));
  }
```

// Compare Student wrt roll

Part B

8.

a) In python list and tuple are a class of data structure that can store one or more object. but they have certain ~~deferances~~ differences -

List	Tuple
(i) It is mutable	(i) It is immutable
(ii) The implication of iterations is time-consuming in the list	(ii) Implecation of iterations are much faster in tuples.
(iii) operations like insertion and deletion are better performed	(iii) Element can be accessed better.
(iv) Consume more memory	(iv) Consume less memory
(v) Many built-in-methodes are available	(v) Does not have many built in methodes.

b)

```
a = int ( input ("Enter first no.: "));
b = int ( input ("Enter second no: "));
try:
    print ("result of division:", (a/b));
except Exception as e:
    print ("This division is not possible!",e);
finally:
    print ("closed");
```

Here in this example if a = any integer and b is any integer except zero then the division is possible but if b = 0 then diversion is not possible. To implement this idea we need exception handling.

Here at first try block executed first but if there any exception occury then it executes except block and at last finally block must executed in each case.

So here if b=0 then Except block is executed and it returns division by zero exception.

9.

Some of the immutable data types in python are int, float, decimal, bool, string, tuple and range, int :

7.

Advantage of python over Java, as OOP:

1) Code is more concise in python
2) The python package index contains numerous third party modules that make python capable of interacting with most of the other languages
3) Dynamic typing is supported in python
4) python has user friendly data structures.
5) python provides enhanced process control capabilities and processes strong integration.

c) python is simpler to ~~use~~ ~~Jython~~ use because it is a scripting language unlike Java (which inherits most complexities form.

7) python provides a large standred library. Many high use programing tasks have already been scripted into the standred library which reduces the length of the code to be written significantly.

10.

```
Class TextSearch:
    @classmethod
    def palindromes(cls, text):
        tokens = text.split()
        Pals = set()
        for words in tokens:
            if word == "".join
                (reverse(word)):
                Pals.add(word)
        return Pals
```

* This palindromes() function returns a sequence of palindromes found in the text.

```
@classmethod
def uniquewords(cls, text):
    """return a sequence of
       unique words in text"""
    tokens = text.split()
    uniques = set()
    for word in tokens:
        uniques.add(word)
    return uniques
```

11. All the member in a python class are public by default, Any member can be accessible from outside the class environment- There are 3 types of Access modifiers -

for a class in python-

(i) Access Modifier :

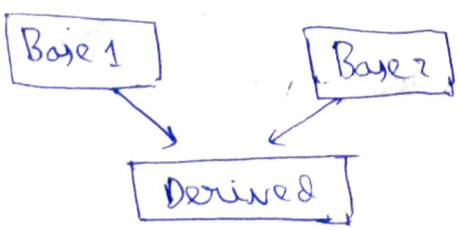
Public - The members declared as public are accessible from outside the class through an object of the class.

protected - protected members are accessible from outside the class but in a class derived from it that is the child or subclass. This variable can be accessed within the same package.

private :- These members are only accessible from within the class. No outside access is allowed.

Multiple inheritance :-

When a class is derived from more than one base class, it is called multiple inheritance. The derived class inherits all the features of the base class.



Example:-

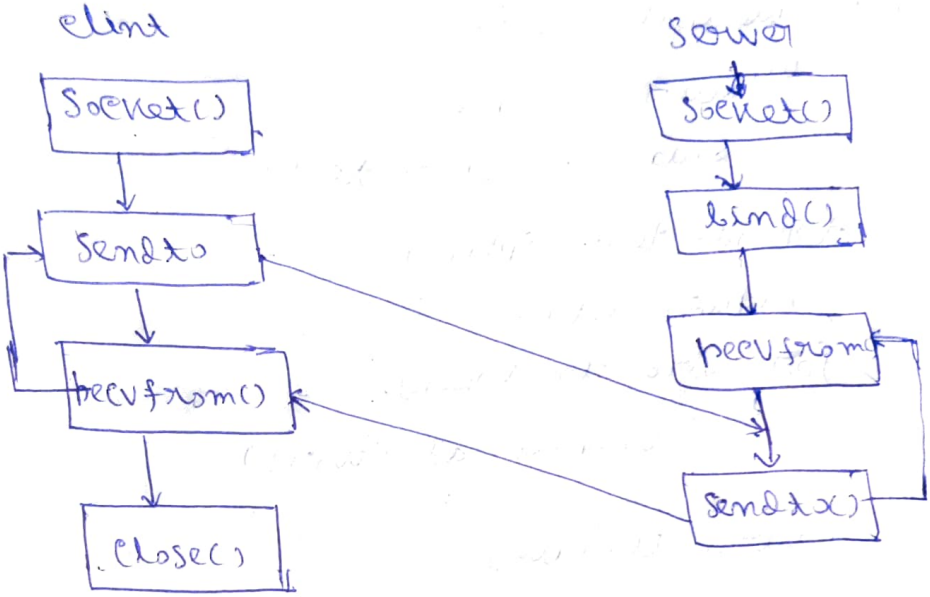
```
class Base1:
    # body of the class

class Base2:
    # body of the class

class Derived(Base1, Base2):
    # body of the class
```

Here all the members of public and private memembers of the classes Base1 and Base2 are accessible at Derived class

12.

```
import socket
Sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
Sock.bind(('127.0.0.1', 12345))
while True:
    dat, addr = Sock.recvfrom(4096)
    print(str(data))
    message = bytes("Hello I am UDP Server").encode('utf-8')
    Sock.sendto(message, addr);
```

UDP protocol -



UDP is connectionless protocol,
It not form a connection with server. Server didn't accept a connection it only waits for a datagram to arrive, datagram contains address of sender which the server want to send data to the correct client. So here no handshaking occure in this case