Java - Week 1

- Explore concepts in PL, Java is an imperative lang. and its oop, design discussions
- Every language has its compromises and understand why we have so many languages out there

Activity Question - 1

1) Which of the following statements is/are true about programming languages?

- [ ] A programming language is a medium for communication between humans.
- [x] A programming language is used to provide computational instructions.
- [x] A programming language is a medium for communication between humans and machines.
- [ ] A program written in a high level programming language is directly executed by the computer.

2) Which of the following statements is/are true about a variable?

- [x] A variable is a name which identifies a memory location.
- [x] A variable is a symbolic representation of the memory location.
- [ ] A variable value cannot be changed once it is initialized.
- [ ] A variable should be initialized during declaration.

3) Which of the following statements is/are true?

- [ ] A compiler converts a program in low level programming language into a code in high level language.
- [x] An interpreter converts a program in high level programming language into code in low level machine language, instruction by instruction.
- [ ] An interpreter converts low level machine language code into high level programming language code, instruction by instruction
- [x] A compiler and an interpreter convert code in high level programming language into low level machine language code.

4) Consider the Python code given below and select the correct option.

```
def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n * recur_factorial(n-1)
```

- ( ) The above code uses imperative programming approach.
- (•) The above code uses declarative programming approach.

5) Consider the Python code given below and select the correct option.

```
def factorial(n):
    if n &lt; 0:
        return 0
    elif n == 0 or n == 1:
        return 1
    else:
        fact = 1
        while(n &gt; 1):
            fact *= n
            n -= 1
        return fact
```

- (•) The above code uses imperative programming approach.
- ( ) The above code uses declarative programming approach.
- ( ) The above code uses both imperative and declarative programming approaches.
- ( ) The above code uses neither imperative nor declarative programming approaches.

6) Which of the following statements is/are correct?

- ( ) Registers are present in memory, and do not allow computation.
- (•) Registers are present in CPU, and allow computation.
- ( ) Registers are present in CPU, and do not allow computation.
- ( ) Registers are present in both memory and CPU, and allow computation.

7) Which programming style may store intermediate values for computation?

- (•) Imperative style of programming
- ( ) Declarative style of programming
- ( ) Neither I nor II
- ( ) Both I and II

- Types has many uses - helps to understand arbitary bit sequences in memory, organize concepts of code nicely, helps compilers to catch bugs early.
- Some lang. support automatic type reference - understand the type of variables from context.

Activity Question - 2

1) Which of the following statements is correct about *static typing*?

- ( ) It is convenient because type of a variable must be declared in advance
- ( ) Bugs cannot be caught early because the type is decided after compilation
- (•) Bugs are caught early because type of variable is declared in advance
- ( ) Type of a variable is not fixed even after variable declaration

2) Which of the following statements is/are correct about *dynamic typing*?

- [x] It is convenient because type of a variable need not be declared in advance
- [ ] Type errors are detected by compiler at compile time
- [x] Type errors are detected at run time
- [ ] Type of a variable is fixed and cannot be changed

3) Which of the following features is/are INCORRECT about *static types*?

- ( ) Helps compiler to catch bugs early
- ( ) Making sense of arbitrary bit sequence in memory
- (•) Makes compiler run faster
- ( ) Organizing concepts in code in a meaningful way

4) Which of the following statements is/are correct regarding the variable declaration in a programming language?

- [x] Variable declaration may result in an error during type checking at the compile time.
- [ ] Variable declaration is a waste of effort and increases the cost by throwing errors at an early stage
- [ ] Variable declaration is a property of dynamically-typed programming language.
- [x] Variable declaration is a property of statically-typed programming language.

5) How to check whether a given program halts?

- ( ) The compiler can check whether a program halts.
- ( ) Alan Turing proposed an algorithm to check if a program halts.
- (•) It is impossible to decide whether a program will ever run indefinitely, or halt.
- ( ) We need to run the program and check.

- Variables have scope (whether var is available in the program) and lifetime (whether the storage is allocated).
- Activation records are kept as a stack, control link links to previous activation records, and return value links tells where to store the result.
- Heap is used to store dynamically allocated data, it outlives the activation record for a func that created the storage, need to be careful about deallocating heap storage - concept of automatic garbage collection

Activity Question - 3

1) The concepts of scope and lifetime are applicable for variables.
- (●) True ✓
- ( ) False

2) Return value link tells where to store the result.
- (●) True ✓
- ( ) False

3) Automatic garbage collection cleans up stack memory.
- ( ) True — no it just cleans the 'not-used' spaces in memory.
- (●) False ✓

4) Which of the following statements is correct? **2 points**
- ( ) Call by value has side effects because it copies the value — ✗ → it can initialized and it pts. to same loca n
- ( ) Function parameters cannot be initialized using call by reference
- ( ) Call by reference has side effects because parameters and arguments refers to different memory location
- (●) Call by reference has side effects because parameters and arguments refers to same memory location ✓

5) Pick the correct statement: **2 points**
- ( ) All active variables are in scope. ✗ not necessarily
- ( ) A variable can not be active and in scope at the same time. ✗
- (●) A variable may be active but not in scope. ✓ → not being used at that moment.
- ( ) None of the above.

6) How does the memory space get allocated during the run time?
- ( ) Memory space is allocated during run time by creating new variables. ✗
- (●) Heap memory is dynamically allocated during run time. ✓ → lecture
- ( ) Stack memory is dynamically allocated during run time.
- ( ) Dynamic memory allocation is not possible.

7) Fill the space with suitable option.
Control link points to the_____
- ( ) Child activation record.
- (●) Parent activation record. ✓ from lecture
- ( ) Current activation record.
- ( ) None of the above.

- Program refinement - focus on code, not much change in data structures
- Modular SD - use refinement to make components, prototype for each component, improve each component while preserving interface and implementation
- Top down - break task into components and then bottom up - combine building blocks
- PL for abstraction - control flow (functions, procedures) and data (abstract dts and oop)

Activity Question - 4

1) Which of the following is/are the feature(s) of having higher modularity in programming?
- [x] Increases code readability – code is easy to understand, and maintainability – code is easy to modify ✓
- [x] Increases code reusability - same code can be reused without duplicating it. ✓ → no it'll less since less amt. of code will be there to debug
- [ ] Increases effort in debugging – identifying and correcting errors require more effort.
- [x] Allows collaborative coding – programmers from the organizations can be coding the same program concurrently. ✓

2) In top-down approach complex task is broken down to sub-tasks
- (●) True ✓ from lecture
- ( ) False

3) Internal data representation should be accessible in data abstraction
- ( ) True
- (●) False ✓ no, it's pvt./encapsulated

- Objects are like abstract datatypes, it helps to encapsulate different combinations of data and functionality
- Abstraction - public interface, pvt implementation, like ADTs
- Subtyping - heirarchy of types, compatibility of interfaces
- Dynamic Lookup - choice of method implementation is determined at run time
- Inheritance - reuse of implementation

Activity Question - 5

1) Match the following.

A. Queues	I. Insertion and deletion possible from both the ends.
B Stacks	II. FIFO
C. Deques	III. LIFO

- ○ A-III, B-II, C-I
- ○ A-II, B-I, C-III
- ◉ A-II, B-III, C-I
- ○ A-I, B-III, C-II

Queue + Stack
↓ ↓
FIFO LI
FIFO

2) An object can be .....

- ○ A single integer ✓
- ○ A file system ✓
- ○ Database ✓
- ◉ All of the above ✓

5) Which of the following statements is/are correct about *subtyping*?

- ☐ Subtyping is same as inheritance ✗
- ☑ If **A** is subtype of **B** then **A** is specialization of **B**
- ☐ If **A** can be implemented using **B** then **A** must be subtype of **B** ✗
- ☑ Subtyping is relationship of interfaces ✓

B → A (it has B + more)

Inheritance is relationship of implementations.

3) Which among the following are features of object-oriented programming language?

- ☑ Dynamic lookup
- ☑ Abstraction
- ☑ Subtyping
- ☑ Inheritance

4 features

4) Which of the following statements is correct about *dynamic lookup*?

- ○ It is the same as method overloading
- ○ It tells whether data member can be accessed on an object
- ◉ It tells us the choice of method implementation at run time → by def^n
- ○ Re-use of implementations is possible because of dynamic lookup → inheritance

6. Match the following —

A. Stack and Queue	I. Supports insert-front() and delete-front()
B. Deque	II. Inherit from Deque → both stack & queue
C. Stack	III. Supports insert-rear() and delete-front()
D. Queue	IV. Subtype of stack and queue

- ○ A-II, B-I, C-IV, D-III
- ○ A-II, B-IV, C-III, D-I
- ◉ A-II, B-IV, C-I, D-III ✓
- ○ A-III, B-IV, C-I, D-II

Stack → I
Queue → III
Deque → II, IV

- Class is a template which describes instance var. and methods for ADT, object is one instance
- Public interface should be seperated from pvt. implementation.
- Heirarchy of classes is impt to implement subtyping and inheritance
- Python has no mechanism for privacy, thus manipulating pvt. instance variables and brings inconsistencies between subtype and parent type.
- Java has declarations like pvt. etc, it is necessary to declare those things and not depend on programmers discipline, it helps to catch bugs early by checking types

Activity Question - 6

1) In Object-Oriented programming languages, object is determined by its ..... 2 poin

- ○ State
- ◉ Class
- ○ Behavior
- ○ Variable

obj. is a concrete instance of a class.

2) Which of the following statements is/are true about inheritance? 2 poin

- ☐ According to inheritance, we can reuse the functionality of one object to define at most one another object. ✗
- ☑ According to inheritance, we can reuse the functionality of one object to define any number of other objects. ✓
- ☐ Inheritance is a relationship of interfaces. → subtyping
- ☑ Inheritance is a relationship of implementations. ✓

3) Consider a simple graphics program to draw different shapes like rectangle, circle and triangle. Which of the following features of Object-Oriented programming language is used to implement this program to perform same operation called draw() on different shapes? 2 points

- ○ Abstraction
- ◉ Dynamic lookup
- ○ Subtyping
- ○ Encapsulation

so when prog. starts running it take relevant shape during implementation

4) Which of the following statements is/are correct? 2 points

- ☑ A class is a template for a data type. ✓
- ☐ A class is a concrete instance of a template ✗ → obj.
- ☐ The instance variables of different objects of the same class will have the same value. → may have diff. values
- ☑ Sending a message to an object refers to the process of asking an object to execute a function on itself. ✓

Practice Assignment - 1

1) Which of the following statements is/are true about the *type of a variable*?

- [ ] In Java, the type of a variable depends on the value assigned to it.
- [x] In Python, the type of a variable depends on the value assigned to it.
- [ ] In Python, the type of a variable can be determined by its data type at compile time itself.
- [x] In Java, the type of a variable can be determined by its data type at compile time itself.

dynamic type

↳ static typing

2) Which of the following statements is true about *dynamic lookup*?

- ( ) Dynamic lookup means different objects respond in the same way to the same message.
- (•) Dynamic lookup means different objects respond in different ways to the same message.
- ( ) Dynamic lookup means the same object responds in different ways to the same message.
- ( ) Dynamic look up hides internal details of an object.

at the run time

the implementation differs

3) Which of the following features of an object-Oriented programming language hides implementation details?

- ( ) Dynamic lookup
- (•) Abstraction → public interface, put. implementation, ADTs
- ( ) Subtyping
- ( ) Inheritance

4) Suppose 'a' and 'b' are two objects such that object 'a' has all the functionalities of object 'b' along with some extra functionalities. Then, which of the following statements is/are true? b → a

- ( ) We can make object 'a' as a subtype of object 'b', and use object 'b' wherever object 'a' is required.
- ( ) We can make object 'b' as a subtype of object 'a', and use object 'b' wherever object 'a' is required.
- (•) We can make object 'a' as a subtype of object 'b' and use object 'a' wherever object 'b' is required.
- ( ) We can make object 'b' as a subtype of object 'a', and use object 'a' wherever object 'b' is required.

5) Which among the following statements is/are true about writing programs in low level languages and high level languages? 2 poi

- [x] Writing a program in a low level language is error-prone.
- [ ] A program written in a low level language is more readable and easy to maintain.
- [ ] A programmer has more control over how a code is mapped to the machine architecture when the code is written in a high level language.
- [x] Language translators like compilers and interpreters are required to convert a program in a high level language into a low level language.

can't write less context

and not in our hands

6) Which among the following statements is/are true about imperative and declarative programming style? 2 poi

- [ ] An imperative program focuses on what the output is and how the output is related to the input, whereas a declarative program focuses on how to obtain the output.
- [x] An imperative program describes the steps of instructions to produce the out-put, whereas a declarative program describes what the output should be.
- [x] An imperative program depends on a large set of instructions that manipulate the intermediate variables, whereas a declarative program may totally avoid using intermediate variables.

how → what

↳ uses inductive structures

7) Which of the following statements is/are true about *variables*?

- [ ] Any variable can be accessed if it is on the stack.
- [x] A variable should be in the correct scope in order to be accessible.
- [ ] Statically and dynamically created data is stored on the stack.
- [x] During the lifetime of a variable, there may be times when it is not in scope.

↳ it can even be out of scope

→ heap

yes, this is possible

8) Let **A** be an object of type **Animal** and **B** be an object of type **Bird**.

```
class Animal:
    def __init__(self, L):
        self.Legs = L
    def walk(self):
        print("Animal walks with", self.Legs, "legs")
class Bird(Animal):
    def __init__(self, L, W):
        self.Legs = L
        self.Wings = W
    def fly(self):
        print("Bird flies")
```

Which of the following options is NOT valid?

- ( ) B.fly( )
- (•) A.fly( )
- ( ) A.walk( )
- ( ) B.walk( )

Animal ↓ Bird → it can walk + it can fly

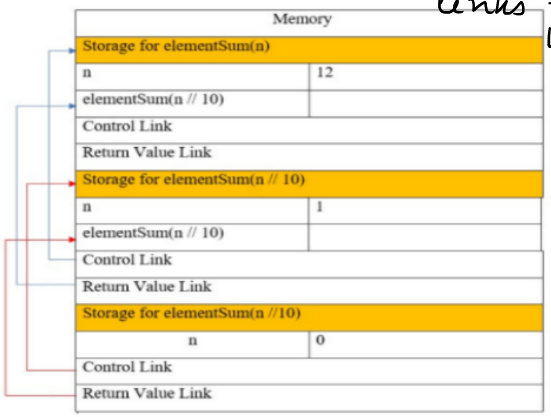
where as animal can't fly

9) Choose the correct memory mapping for the following pseudocode:

```
def elementSum(n):
    if(n == 0):
        return 0
    else:
        return n % 10 + elementSum(n//10)
elementSum(12)
```

control links → pts. to prev. activation record

return value links → tells where to store result



10) Consider the following Python code.

```
def area(radius):
    return(22/7*radius*radius)
def area(breadth,length):
    return(length*breadth)

print(area(7))
print(area(7,10))
```

```
TypeError                                 Traceback (most recent call last)
<ipython-input-1-a234fc2e1068> in <module>
      4         return(length*breadth)
      5
----> 6 print(area(7))
      7 print(area(7,10))

TypeError: area() missing 1 required positional argument: 'length'
```

since the next defn of area overrides the 1st defn.

So the new defn req. 2 arguements, whereas in 1st print statement, 1 arguement is missing

What will be the output of the code?

- (•) The code generates an error
- ( ) 154,70
- ( ) 70,70
- ( ) 154,154

11) From among the following, choose the INCORRECT characteristic/s about Abstract Data Types.

- [ ] Fixed interface
- [ ] Structured data collection
- [x] Fixed implementation
- [ ] Implementation can be changed but should not affect interface

implementations are allowed to δ.

12) Let **A** be an object of **Vehicle, B** be an object of **Twowheeler**. What is the output of **B**.display( )?

```
class Vehicle:
  def display(self):
    print("Every vehicle has wheels")
class Twowheeler(Vehicle):
  def display(self):
    print("Every Two wheeler has two wheels")
```

- Error
- Every vehicle has wheels
- Every Two wheeler has two wheels Every vehicle has wheels
- Every Two wheeler has two wheels

A, B → subtype → both have display() methods.

B.display() — In this ques^n only B is calling the display() method. Hence, it only gets executed.

Graded Assignment - 1

1) Match the following:

A. Control Link	I. Region of the program where a variable is available for use
B. Activation Record	II. Pointer to the previous activation record
C. Scope	III. Duration/time during which a variable is available in the memory
D. Lifetime	IV. Stores the local variables

- A-II, B-I, C-IV, D-III
- A-II, B-IV, C-I, D-III
- A-II, B-IV, C-III, D-I
- A-I, B-IV, C-III, D-II

direct defn; scope; control link; activation record; Life time

2) What will be the output of the following Python code?

```
class Demo:
    def __init__(self,str):
        self.name = str
    def print_Demo(self):
        print(self.name)

obj1 = Demo("IITM")
obj2 = Demo("Java")
obj1.print_Demo()
obj2.print_Demo()
```

- IITM Java
- IITM IITM
- Java Java
- Java IITM

obj1.print_Demo() → self.name; obj2.print_Demo() → self.name

3) What will be the output of the following Python code?

```
i = 42
def f():
    j = i+10
    print(i)
f()
print(j)
```

```
42
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-ff79e9ec12e7> in <module>
      4 print(i)
      5 f()
----> 6 print(j)

NameError: name 'j' is not defined
```

- 42 52
- 42 0
- 42 followed by an error
- Error

j = ? no return of j; only j is local and not global variable

4) Consider the Python code given below and choose the correct option.

```
def fun1(x):
  y = x+1
  def fun2(z):
    z = ((x+y)*(x-y))*z
    print(z)

fun2(2)
```

fun1(5)

- x, y, z are stored in the same activation record.
- x and y both are stored in the same activation record, whereas z is stored in another activation record.
- x, y, z are each stored in different activation records.
- None of the above

func^n 1 ⇒ x, y } 2 activation record
func^n 2 ⇒ z

5) Consider the statements given below.

Statement 1: **Player** is an object that has name, age and role, as its data.
Statement 2: **Captain** is an object that has name, age, role, date of appointment as a captain, and number of years of experience, as its data.

Identify the correct option regarding subtyping with respect to **Player** objects and **Captain** objects.

- Captain can be a subtype of Player.
- Player can be a subtype of Captain.
- Captain cannot be a subtype of Player.
- Player cannot be a subtype of Captain.

Player ↓ subtype Captain

6) Consider writing a 'low level' machine language program to perform the following operation:

$$d = a / (b * c)$$

Identify the correct order to execute the steps given below to perform the given operation.

1. Load the value from memory location a into register R2.
2. Load the values from memory locations b and c into registers R1 and R2 respectively.
3. Subtract the content of R1 from R2 and store the result back into R1.
4. Multiply the contents of registers R1 and R2 and store the result back into R1.
5. Store the content of R1 into memory location d.

- 2->1->4->3->5
- 2->4->1->3->5
- 1->2->4->3->5
- 2->4->5->1->3

b → R1, c → R2; R2 (free); R2 − R1 → R1; R1 → d

7) Identify the most appropriate segregation of the following features between (A) static and (B) dynamic typing in the context of programming.

1. A name in the program derives its data type from the assigned value.
2. Every name needs to be declared with its type in advance.
3. An uninitialized name has no type.
4. A name cannot be assigned to an incompatible value (a value whose type is not compatible with the type of the name).
5. A name can be assigned to any value, and the type of the value determines the type of the name.
6. It does not allow any name to be assigned unless it is already declared explicitly with type.
7. During any assignment, it cannot identify either if it is an assignment for a new name or if it is a reassignment for an existing name.

- ~~(A) - 2, 4, 6~~
  (B) - 1, 3, 5, 7
- (A) - 1, 3, 5
  (B) - 2, 4, 6, 7
- (A) - 1, 3, 4, 5
  (B) - 2, 4, 7
- (A) - 2, 4, 6, 7
  (B) - 1, 3, 5

Static
→ name + types (in adv.)
→ can't have incompatible
→ can't allow, unless explicitly told

and vice-versa for dynamic
since, it is dynamic and reinitialization can take place at any pt. of the program

8) Consider a polyclinic which has a number of doctors. The doctors have schedules for their visiting days and times. Doctors can update their profiles and change their visiting times. Patients need to register in order to seek appointments with the doctors. Doctors provide prescriptions of various medicines to the patients.
Given the above scenario, match the abstract types with the associated set of operations.

- • Doctor type with the operations – add and modify own profile, add and modify visiting timing, seek appointment
  • Prescription with the operation – write prescription
  • Medicine type with the operation – add medicines
- • Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription, seek appointment
  • Patient type
  • Prescription type with the operation – write prescription
  • Medicine type with the operation – add medicines
- • Doctor type with the operations – add and modify own profile, add and modify visiting timing, write prescription
  • Patient type with the operation – seek appointment
  • Prescription with the operation – add medicines
  • Medicine type
- • Doctor type with the operations – add and modify own profile, write prescription
  • Patient type with the operations – seek appointment, add and modify visiting timing,

Doctor → schedule for days & times
→ update profile
→ Δ times
→ provide prescriptions to patients

Patients → register to seek appointment

these 2 are enough to identify from the rest others

9) What will be the value of **num** after execution of the following code?

```
def elementSum(n):
    sum = 0
    while (n != 0):
        sum = sum + n % 10
        n = n // 10
    return sum
num = 22
x = elementSum(num)
```

- 0
- 22
- 4
- 2
- 10

num & n → have diff. memory locns, no change, hence,
so whatever happen it doesn't matter, the ques is asking for 'num'
if x, then value would be 4.

10) What will be the value of **Dict** after execution of the following code?

```
def updateDict(d):
    d["rollno"] = 12
Dict = {"name" : "John", "Age" : 21}
updateDict(Dict)
```

- {"rollno" : 12, "Age" : 21}
- {"rollno" : 12}
- {"name" : "John", "Age" : 21}
- {"name" : "John", "Age" : 21, "rollno" : 12}

dict & d are pting to same memory locns hence, it'll affect the Δ. The rollno. key isn't already ✓, hence it gets added.

Java - Week3

- Algo + data struc = programs, algos come first then data representation
- OOP - reverses the process, first identify data and then later choose which algo will work for it, this is beneficial for large systems
- Noun signify objects, verbs denote method that operate on objects
- State - info in the instance variable, Encapsulation - shouldn't change until a method operates on it, states may be same and they typically affect behaviour
- Relationship btw. classes - dependance, aggregation, inheritance
- Accessor and mutator methods to set instance variables
- If a is subclass of b, then a inherits all methods from b and has some special methods too, how can constructor of a can access pvt instance var of b? by using b class constructor + "super"
- Subclass extends parent class, they inherit instance var and methods, and they can add too.
- They can override methods too, dynamic dispatch - which implementation to call at the run time
- Signature of a function is its name and list of arguement types. In multiple constructors, we can have diff functions with same name and diff signatures.
- Overloading - multiple methods, diff signatures, choice is static
- Overriding - multiple methods, same signatures, choice is static
- Dynamic dispatch - multiple method, same signatures, choice made at run time
- Use type casting and reflection overcome static type restrictions

Activity Question - 1

1) Structured programming divides a program into a set of functions, then combine them to build complex systems.

- [x] True
- [ ] False

top bottom → bottom up process for complex systems

2) Structured programs are easier to debug as compared to object-oriented programs because they allow developing a program with a set of functions without taking care of any objects.

- [ ] True
- [x] False

u have to take care of obj.

3) Object-oriented design is ideal for complex systems.

- [x] True
- [ ] False

yes, oop → 4 features

4) Consider the code given below and select the correct statements.

```
public class Employee{
        private String name;
        private String dept;
        public Employee(String name, String dept) {
            this.name = name;
            this.dept = dept;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getDept() {
            return dept;
        }
        public void setDept(String dept) {
            this.dept = dept;
        }
}
public class Manager extends Employee{
}
```

Employee → Manager

- [ ] private variables of Employee can be changed directly by a Manager object.
- [ ] private variables of Employee can be changed in the Manager class using Accessors.
- [x] private variables of Employee can be changed in the Manager class using Mutators.
- [x] private variables of Employee can be changed in the Manager class using superclass constructor.

no this isn't possible → it helps to reach & not Δ

Activity Question - 2

1) Consider the Java code given below.

```
public class India{
    public India(){
        System.out.println("Visit Northeast India.");
    }
}
public class Northeast extends India{
    public Northeast(){
        System.out.println("Kolkata connects Manipur,
        Assam, Nagaland and Tripura directly by air.");
    }
}
public class Example{
    public static void main(String[] args){
        Northeast charming = new Northeast();
    }
}
```

What will the output be?

- [ ] Visit Northeast India.
- [ ] Kolkata connects Manipur, Assam, Nagaland and Tripura directly by air.
- [x] Visit Northeast India.
  Kolkata connects Manipur, Assam, Nagaland and Tripura directly by air.
- [ ] Compilation failed.

Visit Noeast..

Main

both statements get printed

2) In general, a parent class has more features than a subclass.

- [ ] True
- [x] False

No, vice versa is true.

Parent → a
↓
Subclass → Parent + Sth. more

3) We can create an object of the _____ class by taking reference of the _____ class.

- [ ] parent, child
- [x] child, parent

child parent

4) Which keyword is used to access the constructor of the parent class?

- [ ] extends
- [x] super
- [ ] this
- [ ] parent

direct

5) Which of the following statements is/are true about inheritance? 2 point

- [x] If class A is derived from class B, then A is called the child/subclass and B is called the parent/super class.
- [ ] If class A is derived from class B, then the functionality of class A is added to that of class B.
- [x] If class A is derived from class B, then the functionality of class B is added to that of class A, except those of private and final components.
- [ ] All of the above

6) Consider the statements given below and select the correct option. 2 point

Statement 1: Class A is derived from class B.
Statement 2: Class C is derived from class B.

- ( ) It is not possible to derive class A and C from a single class B.
- ( ) Class C gets the functionalities of both class A and class B.
- (•) Classes A and C are different from each other, but derived from the same class B.
- ( ) All of the above.

8) Which of the following classes is by default the base class of a given class A, if class A is not inherited from any other class?

- (•) Object
- ( ) Thread
- ( ) Throwable
- ( ) None of the above

7) Select the appropriate code snippet to derive class A from class B.

( )
```
class A extend B
{
  //statements
}
```

(•)
```
class A extends B
{
  //statements
}
```

( )
```
class B extend A
{
  //statements
}
```

( )
```
class B extends A
{
  //statements
}
```

9) Consider the Java code given below.

```
public class A{
    int a=10;
    int b=20;
}
public class B extends A{
}
public class Example{
    public static void main(String[] args){
        B ob=new B();
        A oa=new A();
        ob.a=20;
        System.out.println(oa.a);
        System.out.println(ob.a);
    }
}
```

Choose the correct option?

- ( ) Compilation error due to ob.a=20.
- ( ) Compilation error due to System.out.println(oa.a)
- (•) ~~10~~ 20

Activity Question - 3

1) Consider the Java code given below.

```
public class A{
  void show(){
     System.out.println("Show without parameters");
  }
}
public class B extends A{
  void show(int a){
    super.show();
    System.out.println("Show with int parameter");
  }
  public static void main(String args[]){
    B ob=new B();
    ob.show(10);
  }
}
```

What will the output be?

- (•) Show without parameters
Show with int parameter
- ( ) Show with int parameter
Show without parameters
- ( ) Compilation error.

2) Consider the statements given below and select the correct option. 2 point

Statement 1: Methods can be overloaded with different number of parameters.
Statement 2: Methods can be overloaded with different order of parameters (order should be changed along with parameter type).
Statement 3: Methods can be overloaded with different type of parameters.

- ( ) Only statements 1 and 2 are true.
- ( ) Only statements 2 and 3 are true.
- ( ) Only statements 1 and 3 are true.
- (•) Statements 1, 2 and 3 are true.

3) Which of the following statements is/are true about inheritance in Java?

- ( ) Super class constructors can be overloaded in the subclass.
- ( ) Super class constructors can be overridden in the subclass.
- ( ) Super class constructors can be overloaded and overridden in the subclass.
- (•) None of the above.

4) The private components of the parent class are ____ to the subclass.

- ( ) visible
- (•) invisible

5) Answer true/false: In addition to the inherited components, a subclass can add more instance variables and methods.

- (•) True
- ( ) False

6) Answer true/false: Parent class inherits from the subclass.

- ( ) True
- (•) False

7) Match the following

A. Overloading	I. Methods having same name and but belonging to different classes
B. Overriding	II. Calls methods based on the run time identity of the object
C. Dynamic Dispatch	III. Code reusability
D. Inheritance	IV. Methods belonging to same class having same name and different arguments

- ( ) A-I, B-IV, C-IV, D-II
- ( ) A-I, B-III, C-IV, D-II
- ( ) A-IV, B-III, C-I, D-II
- (•) ~~A-IV, B-I, C-II, D-III~~

8) Choose the correct option.

_____ ensures that the most appropriate method is called, based on the run-time identity of the object.

- (•) Dynamic method dispatch

```
public class Animal{
    public void sleep(){
        System.out.println("This animal sleeps everyday....");
    }
}
public class Cat extends Animal{
    public void sleep(){
        System.out.println("Cat sleeps everyday....");
    }
}
public class Dog extends Animal{
    public void sleep(){
        System.out.println("Dog sleeps everyday....");
    }
}
public class Cow extends Animal{
    public void sleep(){
        System.out.println("Cow sleeps everyday....");
    }
}
```

- Can a subclass extend from 2 parent classes? No, java doesn't allow multiple inheritance
- Universal superclass Object, it has some useful methods, we can exploit the tree structure to write generic functions (default like - equals(), toString() - they are implicitely inherited by any class), overriding looks for closest match
- When we override functions, we should be careful to check the signatures.

Activity Question - 4

1) Identify the incorrect statement.

- Parent class can be inherited by multiple subclasses.
- Subclass can extend multiple parent classes.
- Multilevel inheritance is possible in Java.
- The parent class of all the classes is the Object class.

2) Method overriding is one way to achieve polymorphism

- Encapsulation.
- Polymorphism.

3) Answer true/false: Both private and final keywords prevent method overriding.

- True
- False

4) Consider the Java code given below.

```
public class MP3{
    public void translate(String s){
        System.out.print("Afreen"+s);
    }
    public void translate(int i){
        System.out.print("One");
    }
}
public class Play{
    public static void main(String args[]){
        MP3 audio = new MP3();
        audio.translate("Stunning"+" ");
        audio.translate("1");
    }
}
```

What will the output be?

- Afreen One
- Afreen Afreen
- Stunning 1
- Stunning One

5) Consider the Java code given below.

```
public class Animals{
    public void preferTo(){
        System.out.println("Walk.");
    }
}
public class Birds{
    public void preferTo(){
        System.out.println("Fly.");
    }
}
public class Human extends Animals, Birds{
    public void preferTo(){
        System.out.println("Eat, sleep, code and repeat.");
    }
}
public class Example{
    public static void main(String[] args){
        Human obj = new Human();
        obj.preferTo();
    }
}
```

What will the output be?

6) Consider the Java code given below.

```
public class College{
    private String name="IIT Madras";
    public String getName(){
        return name;
    }
    public void setName(String newName){
        this.name = newName;
    }
}
public class Example{
    public static void main(String[] args){
        College obj = new College();
        obj.setName("CMI");
        System.out.println("Pursuing BSC in: "+obj.getName());
    }
}
```

What will the output be?

- [x] The program maintains encapsulation.
- [ ] The output of the program is: Pursuing BSC in: IIT Madras
- [x] The output of the program is: Pursuing BSC in: CMI
- [ ] The program will give compile time error.

7.)

```
public class Animals{
    public Animals(){
        System.out.println("Walk.");
    }
}
public class Birds extends Animals{
    public Birds(){
        System.out.println("Fly.");
    }
}
public class Human extends Birds{
    public Human(){
        System.out.println("Eat, sleep, code and repeat.");
    }
}
public class Example{
    public static void main(String[] args){
        Human obj = new Human();
    }
}
```

What will the output be?

- Eat, sleep, code and repeat.
- Walk.
- Fly.
- Eat, sleep, code and repeat.

- Class heirarchy provides both inheritance (reuse of implementations) and subtyping (compatility of interfaces). Inheritance - subtype can reuse the code of the main type.
- Eg. deque is a subtype of queue and stack, whereas both inherit from deque.
- public/private provide encapsulation of data, static for entities defined inside class that exist w/o creating objects of the class, final whose value can't be changed - class, instance var and methods
- generally, instance var are pvt, accessor and mutator methods are public, static are public, final method can never be overriden by a subclass

Activity Question - 5

1) When it comes to subtyping, the capabilities of the subtype are _____ of/with the main type. **2 points**

- subset
- superset
- disjoint
- not comparable

Main → Subtype] Superset + it includes everything that main type can do

2) Consider the statement: A tree is a connected acyclic graph. **2 points**

Based on the given statement, which among the two - graph and tree - is best modeled as the main type and which as the subtype?

- Graph as the main type and the tree as the subtype.
- Tree as the main type and graph as the subtype.
- Graph and tree cannot have a subtype and main type relationship.
- The answer depends on the specific implementation of graph and tree.

Graph → Tree (defn)

3) Answer true/false: Class hierarchy represents both subtyping and inheritance.

- True
- False

↳ yes it includes both

4) Answer true/false: Deque has more functionality than a stack or a queue. So, queue and stack are subtypes of deque.

- True
- False

subtype of stack & queue and no deque is

Activity Question - 6

1) Which of the following modifiers is used to prevent the inheritance in Java?

- abstract
- final
- public
- All the above

once you put final to method / class / vars. it can't be sed.

2) Consider the Java code given below.

```
public class A{
    public void show(){
        System.out.println("A show() called");
    }
}
public class B extends A{
    void show(){
        System.out.println("B show() called");
    }
}
public class Example{
    public static void main(String[] args){
        B ob=new B();
        ob.show();
    }
}
```

Cannot override → it's like attempting to provide weaker access privileges hence, an error compile time

3) Match the following.

A. public I. Accessor and mutator.
B. private II. No overriding.
C. static III.Instance Variable.
D. final IV. Components that exist without creating objects.

- A-I, B-III, C-IV, D-II
- A-I, B-II, C-IV, D-III
- A-II, B-I, C-IV, D-II
- A-IV, B-I, C-II, D-III

generally, they are public

final

static → they are generally put.

5) To achieve encapsulation, all the methods and instance variables should be declared private.

- True
- False

it's not necessary | both public/put. provide encapsulation

Answer true/false for questions 4 to 6.

4) Access modifiers in Java can be applied to classes, instance variables and methods

- True
- False

6) Modifiers static and final are orthogonal to public or private.

- True
- False

Practice Assignment - 1

1) Consider the following code.

```
public class Example1{
    public int a = 10;
    public void display( ){
        System.out.print("In parent class");
    }
}
public class Example2 extends Example1{
    public int b = 20;
    public void display( ){
        System.out.println("In child class");
    }
    public static void main(String[] args) {
        Example1 obj = new Example2( );
    }
}
```

only eg 2

What is the output of obj.display( )?

- In parent class
- In child class
- Compiler Error
- In parent class
  In child class

display method overridden

2) Consider the following code.

```
class Polygon {
    public void perimeter( ){
        System.out.print("In Polygon perimeter");
    }
    public void angleSum( ) {
        System.out.println("In Polygon angleSum");
    }
}
public class Pentagon extends Polygon{
    public void perimeter( ){
        System.out.println("In Pentagon perimeter");
    }
    public void angleSum(int x) {
        System.out.println("In Pentagon angleSum");
    }
}
```

Same class same signature ↓ then overload

Which method in the parent class has been overloaded in the child class?

- perimeter
- angleSum
- neither
- both

3) Consider the following code.

```
1   public class Animals{
2       final public void swim( ){
3           System.out.println("Animals swim");
4       }
5       public static void communicate( ){
6           System.out.println("Animals communicate");
7       }
8   }
9   public class Birds extends Animals{
10      public void swim( ){
11          System.out.println("Birds swim");
12      }
13      public void communicate( ){
14          System.out.println("Birds communicate");
15      }
16      public void fly( ){
17          System.out.println("Birds fly");
18      }
19  }
```

Identify the line/s which has/have error.

10 & 13 are errors as final & static methods can't be overridden

4) Identify the steps in implementation of:
A. structured programming and
B. object-oriented programming.

algo → data
data → algo

1. Identify the data to be manipulated and maintained
2. Design a set of procedures for specific tasks and combine them to build a complex system
3. Devise / identify algorithms to operate on the data
4. Design data structures to suit procedural manipulations

- A. 1 → 4 B. 2 → 3
- A. 1 → 3 B. 2 → 4
- A. 1 → 2 B. 3 → 4
- (selected) A. 2 → 4 B. 1 → 3 → B

5) Consider the Java code given below.

```
public class Parent{
    public void f1(){
        System.out.println("f1() called from parent class");
    }
}
public class Child extends Parent{
    public void f1(){
        System.out.println("f1() called from child class");
    }
}
public class FClass{
    public static void main(String[] args){
        //LINE 1
    }
}
```

Which of the following statements at LINE 1 will generate the output as:
f1() called from parent class

- (selected) new Parent().f1();
- ((Child)new Parent()).f1();
- new Child().f1();

new Parent() creates an unnamed obj. of parent class, so f1() is called from parent

6)

```
public class Rectangle{
    private final void computeArea(){
        System.out.println("area of rectangle");
    }
}
public class Cube extends Rectangle{
    public final void computeArea(){
        System.out.println("area of cube");
    }
}
public class FClass{
    public static void main(String[] args){
        Cube r = new Cube();
        r.computeArea();
    }
}
```

considered a new method here

What will be the output?

- area of rectangle area of cube
- area of rectangle
- (selected) area of cube

7) Consider the code segment given below. **2 points**

```
public class Data{
    //definition of MAX as constant
}
public class FClass{
    public static void main(String[] arg){
        System.out.println(Data.MAX);
    }
}
```

Consider that MAX is defined as constant in class Data. Identify the most appropriate definition(s) of MAX in class Data such that the output of the code is 100.

- public final int MAX = 100;
- private final static int MAX = 100;
- private static int MAX = 100;
- (selected) public final static int MAX = 100;

final
public is default
constant forever
accessed using class name

8) Consider the Java code given below.

```
public class Employee{
    private int empid;
    private String name;

    public Employee(int empid_, String name_){
        empid = empid_;
        name = name_;
    }
    public Employee(){
        this(0, "unknown");
    }
    public void print()
        System.out.print(empid + " : " + name + " ");
    }
}
public class Manager extends Employee{
    private String department;

    public Manager(int empid_, String name_, String department_){
        department = department_;
    }
    public void print(){
        super.print();
        System.out.println(department);
    }
}
public class FClass{
    public static void main(String[] args){
        Manager m = new Manager(101, "Nutan", "HR");
        m.print();
    }
}
```

What will be the output?

- 0 : unknown
- (selected) 0 : unknown : HR
- HR
- 101 : Nutan : HR

9) Consider the following code.

```
public class A{
    public float g;

    public A(){
        g = 9.8f;
    }
    public void show(){
        System.out.println("g = "+g);
    }
}
public class B extends A{
    public double e;

    public B(double num){
        e = num;
    }
    public void show(){
        System.out.println("e = "+e);
    }
}
public class FClass{
    public static void main(String args[]){
        A obj1 = new B(2.718);
        A ptr1 = (A)obj1;
        A ptr2 = (B)obj1;
        B ptr3 = (B)obj1;

        ptr1.show();
        ptr2.show();
        ptr3.show();
    }
}
```

Upper caste

it is type casting but the obj remains B in all cases

What will be the output of the given code?

- g = 9.8 g = 9.8 e = 2.718
- g = 9.8 e = 2.718 e = 2.718
- (selected) e = 2.718 e = 2.718 e = 2.718

10) Consider the following code.

```
1.  public class Mammal{
2.      public String name;
3.      public int lifespan;
4.
5.      public Mammal(){
6.          name = "Giraffe";
7.          lifespan = 45;
8.      }
9.
10.     public void show(){
11.         System.out.println("Giraffe");
12.         System.out.format("name = %s : lifespan = %d",name,lifespan);
13.     }
14. }
15.
16. public class Endangered extends Mammal{
17.     public boolean endanger_status;
18.
19.     public Endangered(){
20.         endanger_status = false;
21.     }
22.
23.     public void show(){
24.         System.out.println("endanger status of "+this.name
                                + " is "+endanger_status);
25.     }
26.     public void display(){
27.         this.show();
28.     }
29. }
30.
31. public class FClass{
32.     public static void main(String args[]) {
33.         Mammal m1 = new Endangered();
34.         m1.show();
35.         m1.display();
36.     }
37. }
```

m1 is in obj. of mammal and hence, m1.display() will give error

11) What will be the output of the following Java program?

```
public class Numbers{
    public int x = 1;
    public double y = 2.1;

    public Numbers(int a, double b){
        add(a,a);
        add(b,a);
    }
    public void add(int a, int b){
        this.x = a + b;
    }
    public void add(double c, int d){
        this.y = c + d;
    }
}
public class Example{
    public static void main(String args[]){
        Numbers obj = new Numbers(2,3.2);
        System.out.println(obj.x + " " + obj.y);
    }
}
```

add(a,a) → x
add(b,a) → y

int a
double b

a+a = 4
b+a = 5.2

⇒ 4 5.2

12) Consider the Java program given below and select the correct statement.

```
public class A{
    private int a=10;
    private int b=20;
}
public class B extends A{
}
public class Example{
    public static void main(String[] args){
        B ob=new B();
        A oa=new A();
        ob.a=20; // Line 1
        System.out.println(oa.a); // Line 2
        System.out.println(ob.a); // Line 3
    }
}
```

- Illegal access of variable a at Line 1.
- Illegal access of variable a at Line 2.
- Illegal access of variable a at Line 3.
- All of the above.

13) Consider the statements given below and select the correct option. 2 poin

Statement 1: If the same static method exists in both super class and subclass, then the subclass method hides the super class method.
Statement 2: Both the variables and the methods in a super class can be overridden in a subclass.

- Both statements are true.
- Both statements are false.
- Statement 1 is true and Statement 2 is false.
- Statement 1 is false and Statement 2 is true.

14)

```
public class A{
    public static void print(){
        System.out.println("Class A print");
    }
    public void show(){
        System.out.println("Class A show");
    }
}
public class B extends A{
    public static void print(){
        System.out.println("Class B print");
    }
    public void show(){
        System.out.println("Class B show");
    }
}
public class MyClass{
    public static void main(String args[]){
        A oa=new B();
        oa.print();
        oa.show();
    }
}
```

- Class A print
  Class B show

15) What will be the output of the following Java program?

```
public class College{
    public void name(){
        System.out.println("IIT Madras.");
    }
}
public class Example extends College{
    public void name(){
        super.name();
        System.out.println("University of Calcutta.");
    }
    public static void main(String[] args){
        Example obj = new College();
        obj.name();
    }
}
```

- Compilation fails.
- IIT Madras.
- University of Calcutta.

16 - 18

```
public class A{
    public String features(){
        return "Write Once, Run Anywhere.";
    }
}
public class B extends A{
    public String features(){
        return "Object Oriented.";
    }
}
public class C extends A{
    public String features(){
        return "Multithreaded.";
    }
}
public class Example{
    public static void main(String args[]){
        A obj1 = new A();
        A obj2 = new B();
        A obj3 = new C();
        System.out.print(obj1.features()+"\n"
        +obj2.features()+"\n"+obj3.features());
    }
}
```

16) The above program is an example of

- Dynamic dispatch.
- Compile time polymorphism.

17) The output of the above program is:

Write Once, Run Anywhere.
~~Write Once, Run Anywhere.~~
~~Write Once, Run Anywhere.~~

- True
- False

18) What is the return type of the overridden method?

- void
- String

19)

```
public class A{
    public void show(){
        System.out.println("A class show()");
    }
}
public class B extends A{
    public void show(){
        System.out.println("B class show()");
    }
}
public class C extends B{
}
public class Example{
    public static void main(String[] args){
        C oc=new C();
        oc.show();
    }
}
```

- Compilation failed
- Nothing will be printed
- A class show()
- B class show()

20) Consider the Java program given below and pr

```
public class A{
    public A(){
        System.out.println("A() called");
    }
}
public class B extends A{
    public B(){
        System.out.println("B() called");
    }
}
public class C extends B{
    public C(){
        System.out.println("C() called");
    }
}
public class Example{
    public static void main(String[] args){
        C oc=new C();
    }
}
```

21) What will be the output of the following Java program?

```
public final class Music{
    public void artist(){
        System.out.println("Anupam Datta");
    }
}
public class Poet extends Music{
    public void artist(){
        super.artist();
        System.out.println("Rabindranath Das");
    }
}
public class Example{
    public static void main(String args[]){
        Music obj = new Poet();
        obj.artist();
    }
}
```

Graded Assignment - 1

1)

```
public class A {
    public void display( ){
        System.out.print("Hii ");
    }
}
public class B extends A{
    public void display(String s){
        display();
        System.out.println(s);
    }
    public static void main(String[] args) {
        A a = new B();
        // Line 1
    }
}
```

What is the correct instruction to be written in Line 1 in order to print Hii Ram as output?

- a.display("Ram");
- a.display("Hii Ram");
- ((A)a).display("Hii Ram")
- ((B)a).display("Ram");

```
class Polygon {
    public void perimeter( ){
        System.out.print("In Polygon perimeter");
    }
    public void angleSum( ) {
        System.out.println("In Polygon angleSum");
    }
}
public class Pentagon extends Polygon{
    public void perimeter( ){
        System.out.println("In Pentagon perimeter");
    }
    public void angleSum(int x) {
        System.out.println("In Pentagon angleSum");
    }
}
```

2) Which method/s override/s methods in the parentclass?

- perimeter
- angleSum
- neither

3) Consider the following code:

```
1  public class Employee {
2      public void display( ){
3          System.out.print("In Employee class");
4      }
5  }
6  public class TeamLead extends Employee{
7      public void display( ){
8          System.out.println("In TeamLead class");
9      }
10 }
11 public class Manager extends Employee, TeamLead{
12     public void display( ){
13         System.out.println("In Manager class");
14     }
15     public static void main(String[] arg) {
16         TeamLead t = new Employee( );
17     }
18 }
```

Identify the line/s which has/have error.

4) Consider the Java code given below.

```
public class Bird{
    public void fly(){
        System.out.println("it can fly");
    }
}
public class Duck extends Bird{
    public void swim(){
        System.out.println("it can swim");
    }
}
public class FClass{
    public static void dolt(Bird b){
        b.fly();
        if(b instanceof Duck)
            ((Duck) b).swim();
    }
    public static void main(String[] args){
        Duck d = new Duck();
        dolt(d);
    }
}
```

5) Consider the Java code given below.

```
public class Shape{
    public void area(){
        System.out.println("area is unknown");
    }
    public void volume(){
        System.out.println("volume is unknown");
    }
}
public class Rectangle extends Shape{
    public void area(){
        System.out.println("area of Rectangle");
    }
}
public class Cube extends Shape{
    public void area(){
        System.out.println("area of Cube");
    }
    public void volume(){
        System.out.println("volume of Cube");
    }
}
public class FClass{
    public static void compute(Shape s){
        s.area();
        s.volume();
    }
    public static void main(String[] args){
        Rectangle r = new Rectangle();
        Cube c = new Cube();
        compute(r);
        compute(c);
    }
}
```

What will the output be?

- area is unknown
volume is unknown
area is unknown
volume is unknown
- area of Rectangle
area of Cube
volume of Cube
- area of Rectangle
volume is unknown
area of Cube
volume of Cube

6) Consider the following abstract types.

- Scanner, with method scanDocuments().
- Printer, with method printDocuments().
- Copier, with methods scanDocuments() and printDocuments().

Identify the correct subtype and inheritance relationships between the classes.

- Scanner and Printer are subtypes of Copier
- Copier is a subtype of Scanner and Printer
- Scanner and Printer both inherit from Copier
- Copier inherits from both Scanner and Printer

7) Consider the Java code given below.

```
public class Employee{
    private int empid;
    private String name;

    public Employee(int empid_, String name_){
        empid = empid_;
        name = name_;
    }
    public Employee(){
        this(0, "unknown");
    }
    public void print(){
        System.out.print(empid + " : " + name + " : ");
    }
}
public class Manager extends Employee{
    private String department;

    public Manager(int empid_, String name_, String department_){
        //line:1
        department = department_;
    }
    public void print(){
        super.print();
        System.out.println(department);
    }
}
public class FClass{
    public static void main(String[] args){
        Manager m = new Manager(101, "Nutan", "HR");
        m.print();
    }
}
```

Identify the appropriate option to fill in the blank at line:1 such that output of the code will be:

101 : Nutan : HR

- Employee(empid_, name_);
- this(empid_, name_);
- super(empid_, name_);
- empid = empid_; name = name_;

8) Consider the following code.

```
public class Student{
    public String sname;
    public String sid;
    public int sclass;

    public Student(String s_name,String s_id, int s_class){
        this.sname = s_name;
        this.sid =s_id;
        this.sclass = s_class;
    }
    public void display() {
        System.out.println("name:"+sname);
        System.out.println("id:"+sid);
        System.out.println("class:"+sclass);
    }
}
public class Toppers extends Student{
    public int marks;

    public Toppers(int marks){
        this.marks = marks;
    }

    public void display() {
        super.display();
        System.out.println("marks:"+marks);
    }
}
public class FClass{
    public static void main(String[] args){
        Toppers t = new Toppers(30);
        t.display();
    }
}
```

Choose all the correct option/s which mention/s the required modification (if any) in the code for successful compilation.

- [ ] super.display must be removed.
- [x] Define an appropriate no argument constructor of Student class.
- [x] Call the Student class parameterized constructor explicitly inside Topper class's cons
- [ ] No modification required, code compiles successfully.

9)

```
1.  public class Mammal{
2.      public String name;
3.      public int lifespan;
4.
5.      public Mammal(){
6.          name = "Tiger";
7.          lifespan = 45;
8.      }
9.
10.     public void show() {
11.         System.out.format("name = %s : lifespan = %d",name,lifespan);
12.     }
13.     public void display() {
14.         System.out.println("Mammal details");
15.     }
16. }
17.
18. public class Endangered extends Mammal{
19.     public boolean endanger_status;
20.
21.     public Endangered(){
22.     endanger_status = false;
23.     }
24.
25.     public void show() {
26.         System.out.println("endanger status of "+
                    this.name + " is " + endanger_status);
27.     }
28.     public void display(String species) {
29.         System.out.println("Endanger status");
30.     }
31. }
32.
33. public class FClass{
34.         public static void main(String args[]) {
35.             Mammal m1 = new Endangered();
36.             m1.show();
37.             m1.display();
38.         }
39.     }
```

Choose the correct option regarding the given code.

- ( ) This code produces a compilation error on line 37
- ( ) This code produces a compilation error on line 36
- ( ) This code will generate the output:
  endanger status of Tiger is false
  Endanger status
- (•) This code will generate the output:
  endanger status of Tiger is false
  Mammal details
- ( ) This code will generate the output:
  endanger status of *null* is false
  Mammal details

Java - Week 4

- we want to have a class Shape, and Circle, Square, and Rectangle extends shape, better soln
- provide abstract defn of perimeter in shape, this forces subclasses to provide a concrete implementation, remember - put class also as abstract, we cannot create objects from class which has abstract functions
- can still declare variables, whose type is an abstract class, use abstract class to specify generic properties, An interface is an abstract class with no concrete componenets.
- A class that extends the interface is said to implement it. We can extend only 1 class, but can implement multiple interfaces.

Activity Question - 1

1) Consider the Java code given below.

```
public abstract class Calculator{
    abstract int addition(int a,int b);
    abstract int subtraction(int a,int b);
    abstract int multiplication(int a,int b);
    abstract int division(int a,int b);
}
public class Operation extends Calculator{
    public int addition(int c,int d){
        return c+d;
    }
    public int subtraction(int c,int d){
        return c-d;
    }
    public int division(int c,int d){
        return c/d;
    }
    public static void main(String args[]){
        Operation obj=new Operation();
        System.out.println(obj.division(5,2));
    }
}
```

Choose the correct option.

- This program generates the output: 2
- This program generates the output: 2.5

2)

```
public abstract class A{
    final static int current_sem(){
    return 4;
    }
    abstract String sem_subjects();
}
public class B extends A{
    public String sem_subjects(){
        return "DBMS, DSA and Programming concept using Java.";
    }
    public static void main(String args[]){
        A obj=new B();
        System.out.println("Current Semester: "+A.current_sem()
                +"\nSubjects: "+obj.sem_subjects());
    }
}
```

Choose the correct option.

- Compile time error.
- Run time error.
- The output of the program is:
  Current Semester: 4
  Subjects: DBMS, DSA and Programming concept using Java.

3) Consider the Java code given below.

```
public abstract class NewYear {
    final abstract String resolution();
}
public class Year_2022 extends NewYear{
    public String resolution() {
        return "Walk up early"+"\n"+"Do exercise"+"\n"+"Take shower everyday";
    }
    public static void main(String args[]) {
        System.out.print(new Year_2022().resolution());
    }
}
```

Choose the correct option.

- Compile time error.
- Run time error.

4) Consider the Java code given below.

```
public abstract class Maths {
    int num = 1;
    int store = 0;
    public String find_factorial(int y) {
        this.store = y;
        while(y != 0){
            this.num = num*y;
            y = y-1;
        }
        return "The factorial of "+store+" is: "+num;
    }
}
public class Factorial extends Maths{
    public static void main(String args[]) {
        System.out.print(new Factorial().find_factorial(1));
    }
}
```

Choose the correct option.

- Run time error.
- The output of the program is:
  The factorial of 0 is 1
- The output of the program is:
  The factorial of 1 is 1

5) Consider the program given below and choose the correct option.

```
public class Arithmetic{
    public Arithmetic(){
    System.out.println("Basic operators in Mathematics: "); }
    abstract String operations();
}
public class Algebra extends Arithmetic{
    public String operations(){
        return "+, -, x, %";
    }
    public static void main(String args[]){
        System.out.println(new Algebra().operations());
    }
}
```

- The output of the program is:
  Basic operators in Mathematics:
  +, -, x, %
- The output of the program is:
  +, -, x, %
- Compilation fails because Arithmetic is not abstract.
- Run time error.

6) Which of the following keywords is used to create an abstract class?

- final
- abstracts
- abstract
- abstractable

7) Consider the statements given below and select the correct option.
statement 1: Abstract class must contain only abstract methods.
statement 2: Abstract class may also contain concrete methods.

- Only statement 1 true.
- Only statement 2 is false.
- Statement 1 is false and 2 is true

8) Which of the following code snippet is/are valid?

- [x]
```
public abstract class A {
    //no methods
}
```
- [x]
```
public abstract class A {
    abstract void print();
}
```
- [x]
```
public abstract class A{
    void print(){
        System.out.print("This is concrete method");
    }
}
```
- [x]
```
public abstract class A {
    abstract void display();
    void print(){
        System.out.print("This is concrete method");
    }
}
```

9) Consider the Java code given below.

```
public abstract class Ipl{
    public abstract void team(String name);
}
public class CSK extends Ipl{
    public void team(String name) {
    System.out.println(name+" is CSK team");
    }
}
public class SRH extends Ipl{
    public void team(String name) {
        System.out.println(name+" is SRH team");
    }
}
public class Player{
    Ipl ipl;
    String name;
    public Player(Ipl ipl,String name) {
    this.ipl = ipl;
    this.name = name;
    }
    public void team(){
    ipl.team(name);
    }
}
public class Example {
    public static void main(String[] args) {
    Player player1=new Player(new CSK(), "Ravi Jadeja");
    player1.team();
    Player player2=new Player(new SRH(), "Manish Pandey");
    player2.team();
    }
}
```

Choose the correct option.

- Compilation failed because of abstract class cannot be a member of another class.
- Compilation failed because Player constructor will not accept CSK and SRH class objects.
- This program generates the output:
  Ravi Jadeja is CSK team
  Manish Pandey is SRH team
- This program generates the output:
  null is CSK team
  null Pandey is SRH team

10) Is there is any error in the following code?

```
1:  public abstract class AbstractTest{
2:      public static void main(String args[]) {
3:          System.out.println("Testing abstraction...");
4:      }
5:      abstract final void getValue();
6:      void setValue(){}
7:  }
```

- No, the code is correct, and it will print 'Testing abstraction...'.
- Yes, line 1 is incorrect because an abstract class cannot have the modifier public.
- Yes, line 5 is incorrect because: illegal combination of modifiers: abstract and final

11) Consider the Java code given below. 2 poin

```
abstract class Car{
  boolean isSedan();
}
public class Alto extends Car{
  public boolean isSedan(){
    return false;
  }
  public static void main(String[] args){
    Car car = (Car)new Alto();
    System.out.println(car.isSedan());
  }
}
```

Choose the correct option.

- It compiles and runs without any error.
- The code will not compile because the method isSedan() is not declared abstract.
- The code will not compile because typecasting an object of Alto to Car is not permitted.
- The code will not compile because car.isSedan() is not permitted as isSedan() method is not defined inside Car class.

- An interface is a purely abstract class
  - All methods are abstract
- A class implements an interface
  - Provide concrete code for each abstract function
- Classes can implement multiple interfaces
  - Abstract functions, so no contradictory inheritance
- Interfaces describe relevant aspects of a class
  - Abstract functions describe a specific "slice" of capabilities
  - Another class only needs to know about these capabilities

- Static functions can't access instance variables, default functions can be overriden, the conflicts in multiple inheritance comes back, whenever there is a conflict between superclass and interface - the class wins

Activity Question - 2

1) Which of the following keywords is/are used to create an interface?

- final
- interface
- abstract
- inter

2) Which keyword is used to implement an interface?

- implement
- interface
- abstract
- implements

3) Consider the Java code given below.

```
public interface A{
    public String getCoordinates();
}
public interface B{
    public String tellDirection();
}
public class C implements A,B {
    public String getCoordinates(){
        return "41 degrees, 56 minutes, 54.3732 seconds north.";
    }
    public String tellDirection(){
        return "Start; Turn Left, walk: 100m.";
    }
    public static void main(String args[]){
        System.out.println(new B().tellDirection());
    }
}
```

Choose the correct option.

- Compile time error.
- Run time error.

4) Consider the Java code given below.

```
public class Fibo{
  int fibonacci(int x){
    int a=0, b=1, c=0;
    for(int i=1;i<=x-2;i++){
      c=a+b;
      a=b;
      b=c;
    }
    return c;
  }
}
public interface Statement{
  abstract String display();
}
public class Example extends Fibo implements Statement{
  public String display(){
    return "The required fibonacci number is:";
  }
  public static void main(String args[]){
    System.out.println(new Example().display()+" "+new Fibo().fibonacci(7));
  }
}
```

- Run time error.
- The output of the program is:
  The required fibonacci number is: 5
- The output of the program is:
  The required fibonacci number is: 8

9) Consider the Java code given below.

```
public interface A{
    int a=10;
}
public interface B{
```

5) Consider the program given below and choose the correct option to invoke the goAhead method.

```
public interface A{
    default String goAhead(){
        return "Started.";
    }
    abstract String stop();
}
public class B implements A{
    public String stop(){
        return "Break Pressed.";
    }
    public static void main(String args[]){
        //Invoke the method here
    }
}
```

- A.goAhead();
- new B().goAhead();
- new A().goAhead();

7) What is a marker interface in Java?

- It should contain only default methods in its body.
- It should contain only static methods in its body.
- It does not contain any methods in its body.
- It should contain only abstract methods in its body.

it is an empty interface with no fields or methods

6) Consider the statements given below and select the valid code snippet.

Statement 1: A is an interface.
Statement 2: B and C are the classes.
Here, C should be derived from class B and implemen

- public class C implements B,extends A{ //statements }
- public class C implements A,B{ //statements }
- public class C extends A,B{ //statements }
- public class C extends B,implements A{ //statements }

8) Consider the Java code given below.

```
public interface A{
    int a;
}
public interface B{
    int b=20;
}
public class C implements B{
}
public class Example{
    public static void main(String[] args) {
        C oc=new C();
        System.out.println(oc.b);
    }
}
```

since not initialized it is 0

Output → 20

Choose the correct option.

- Compilation fails
- Nothing will be printed
- This program generates the output: 0

9) Consider the Java code given below.

```
public interface A{
    int a=10;
}
public interface B{
    int b=20;
}
public class C implements A,B{
}
public class Example{
    public static void main(String[] args) {
        C oc=new C();
        System.out.println(oc.a);
        System.out.println(oc.b);
    }
}
```

Output → 10 → 20

10) Consider the Java code given below.

```
public interface A{
    int a=10;
}
public interface B{
    int b=20;
}
public class C implements A,B{
}
public class Example{
    public static void main(String[] args){
        C oc=new C();
        oc.a=5;
        System.out.println(oc.a);
        oc.b=15;
        System.out.println(oc.b);
    }
}
```

Can't assign values to final vars.

Choose the correct option.

- Compilation failed.
- Nothing will be printed.

11) Consider the Java code given below.

```
public interface Areas{
    void area(float a,float b);
}
public class Triangle implements Areas{
    public void area(float a,float b){
        System.out.println("Triangle area is "+(0.5*a*b));
    }
}
public class Rect implements Areas{
    public void area(float a,float b){
        System.out.println("Rectangle area is "+(a*b));
    }
}
public class Example{
    public static void main(String[] args) {
        Areas triangle=new Triangle();
        Areas rectangle=new Rect();
        triangle.area(1.3f,2.4f);
        rectangle.area(5.4f,1.25f);
    }
}
```

Code is correct so right values acc. to the formula will be calculated & displayed

12) Consider the requirements given below for an interface declaration for the interface.

Name of the interface: **Behavior**
Methods in the interface: **walk(), fly(), swim().**
Two classes **Humans** and **Parrots** must implement the inte
the **Humans class** implements/overrides the methods **wall**
Parrots class implements/overrides the method **fly()** only.

```
public interface Behavior{
    public default void walk(){}
    public default void swim(){}
    public default void fly(){}
}
```

- An instance variable can be a user defined type. Objects of private class can see private components of enclosing class. Pvt classes provide additional degree of encapsulation.

Activity Question - 3

1) Which of the following statements is true?

- We can define a private inner class inside another class.
- We can define a static inner class inside another class.
- We can define a local inner class inside another class.
- All of the above.

all are posible at relevant places

2) Consider the statements given below and identify the correct option.
Statement 1: Employee and Date are two classes.
Statement 2: Date class should be available to Employee class only.

Employee has a Date

- We should define a private Date class inside Employee class.
- We should define a private Employee class inside Date class.
- Statement 2 is not possible.

3) Statement: Declaring a class as private is not useful because a private class cannot be instantiated. **2 points**

Choose the correct option regarding the given statement.

- The given statement is correct because if a class is declared as private, then we cannot make an object of the class and hence it is not usable.
- The given statement is correct because if a class is declared as private, then its instance variables cannot be initialized.
- The given statement is incorrect because even if a class is declared as private, it can be instantiated from any another class.
- The given statement is incorrect because a class declared as private can be instantiated if it is defined as an inner class.

(True)

- Encapsulation is impt in oop, internal data is pvt, we can access that using public accessor and mutator methods - interaction with state
- Interface describes the capability of objects.

Activity Question - 4

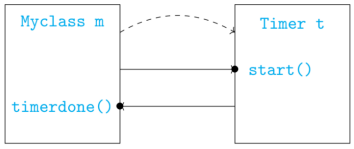
1) Consider the Java code given below.

```
public class Test{
  private int a;
  private int b;
  public int getA() {
    return a;
  }
  public int getB() {
    return b;
  }
  public void set(int a,int b) {
    this.a=a;
    this.b=b;
  }
}
public class ExampleProgram {
  public static void main(String[] args) {
    Test program=new Test();
    program.set(10, 20);
    System.out.println(program.getA());
    System.out.println(program.getB());
  }
}
```

Choose the correct option.

- This program generates the output: 0 0
- Compilation failed.
- Not possible to set values using set(int a, int b).
- This program generates the output: 10 20 ✓

- `Myclass m` creates a `Timer t`
- Start `t` to run in parallel
  - `Myclass m` continues to run
  - Will see later how to invoke parallel execution in Java!
- `Timer t` notifies `Myclass m` when the time limit expires
  - Assume `Myclass m` has a function `timerdone()`



- Timer t should know whom to notify.
- Default interface Runnable indicates that Timer can run in parallel.
- We can use Java class heirarchy to create a generic timer.
- Callbacks are useful when we spawn a class in parallel.

Activity Question - 5

1) Consider the Java code given below.

```
public class A implements Runnable{
    public void run(){
        System.out.println("t2 thread");
    }
}
public class Example{
    public static void main(String args[]){
        A obj=new A();
        Thread t2=new Thread();
        t2.start();
        System.out.println("Hello");
    }
}
```

Choose the correct option.

- Compile time error
- Run time error
- This program generates the output: t2 thread / Hello
- This program generates the output: Hello ✓

code is right. → this won't come since run() has not been called.

- A generic linear list - implementation - array, linked list; we want a loop to run through all the values in a linear list. We create an iterator object and export it.
- Iterator is an example of interaction with state.

```
for (type x : a)
  do something with x;
}
```

for-loop iterator

Activity Question - 6

1) Match the following.

A. Abstract method	1. It can be used to loop through collections.
B. Interface	2. It allows an additional degree of encapsulation. → put. class
C. Iterator	3. It should be overridden. → necessary by child class
D. Private class	4. It cannot be initialized.

- A-3, B-4, C-1, D-2 ✓
- A-3, B-2, C-1, D-4
- A-4, B-3, C-2, D-1
- A-4, B-3, C-1, D-2

2)

```
import java.util.*;
public class Example{
    public static void main (String [] arg){
        ArrayList list = new ArrayList();
        String names[] = {"Ram", "Shyam", "Henry", "Joker", "Mocker", "Locker"};
        for (int i=1; i<names.length; i+=2){
            list.add(names[i]);
        }
        Iterator i = list.iterator();
        while (i.hasNext()){
            System.out.println(i.next());
        }
    }
}
```

- Ram
  Henry
  Mocker
- Shyam
  Joker
  Locker
- Shyam
  Joker

3) Consider the program given below and predict the output.

```
import java.util.*;
public class Example{
    public static void main(String args[]){
        ArrayList<String> str=new ArrayList<String>();
        str.add("Joker");
        str.add("Locker");
        for(int i:str){
        System.out.println(i);
        }
    }
}
```

- Compile time error.
- Run time error.
- Joker
  Locker
- 2

Practice Assignment - 1

1) Consider the code given below.

```
public interface Shape{
    public double area();
    public default double volume() {
        return -1.0;
    }
}

public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public class Rectangle implements Shape, Printable{
    private double w, h;
    public Rectangle(double w_, double h_) {
        w = w_;
        h = h_;
    }
    public double area() {
        return w * h;
    }
    public void print() {
        System.out.print(area() + " ");
        System.out.print(volume());
    }
}

public class FClass{
    public static void main(String[] args) {
        Rectangle r = new Rectangle(20.0, 50.0);
        r.print();
    }
}
```

What will be the output?

- 1000.0 followed by a runtime error
- not implemented
- 1000.0 1.0
- It generates compiler error

2) Consider the code given below.

```
public interface Flyable{
    public void fly();
    public default void travel() {
        System.out.println("travel with wings");
    }
}

public interface Movable{
    public void move();
    public default void travel() {
        System.out.println("travel with legs");
    }
}

public class Bird implements Flyable, Movable{
    public void fly() {
        system.out.println("fly with wings");
    }
    public void move() {
        System.out.println("move with legs");
    }
}

public class FClass{
    public static void main(String[] args) {
        Bird b = new Bird();
        b.fly();
        b.move();
        b.travel();
    }
}
```

What will be the output?

- fly with wings
  move with legs
- fly with wings
  move with legs
  travel with wings
- fly with wings
  move with legs
  travel with legs
- It generates a compiler error

3) Consider the code given below.

```
public interface Comparable{
    public abstract int comp(Comparable x);
}
public class Name implements Comparable{
    private String fname;
    private String lname;

    public Name(String fname, String lname){
        this.fname = fname;
        this.lname = lname;
    }
    public int comp(Comparable x) {
        if(lname.compareTo(((Name)x).lname) == 0)
            return fname.compareTo(((Name)x).fname);
        return lname.compareTo(((Name)x).lname);
    }
    public void print() {
        System.out.println(fname + " " + lname);
    }
}
public class FClass{
    public static void sort(Comparable[] names) {
        for(int i = 0; i < names.length - 1; i++) {
            for(int j = 0; j < names.length - i - 1; j++) {
                if(names[j].comp(names[j + 1]) > 0) {
                    Comparable tname = names[j];
                    names[j] = names[j + 1];
                    names[j + 1] = tname;
                }
            }
        }
    }
    public static void main(String[] args) {
        Name[] names = new Name[] {new Name("Charlotte", "Brown"),
                                   new Name("Ava", "Smith"),
                                   new Name("Emma", "Williams"),
                                   new Name("Olivia", "Smith"),
                                   new Name("Emma", "Johnson")};
        sort(names);
        for(int i = 0; i < names.length; i++)
            names[i].print();
    }
}
```

What will be the output?

- Charlotte Brown
  Emma Johnson
- Ava Smith
  Olivia Smith
  Emma Williams

4) Consider the code given below.

```
1 public abstract class Polygon{
2    public abstract int perimeter();
3 }
4 public interface Shape{
5    public abstract int area();
6 }
7 public class Rectangle extends Shape implements Polygon{
8    public int area( ){
9       System.out.println("length*breadth");
10   }
11   public int perimeter( ){
12      System.out.println("length + breadth");
13   }
14 }
```

Identify the error in the code.

- [ ] Line 2: perimeter( ) is not defined
- [ ] Line 5: area( ) is not defined
- [x] Line 7: incorrect usage of extends and implements
- [ ] Line 7: class Rectangle is not declared as abstract

5) Consider the program given below and choose the correct option.

```
public abstract class NewYear {
    abstract String resolution();
    public NewYear(){
        System.out.println("Resolution: ");
    }
}
public class Year_2022 extends NewYear{
    public String resolution() {
        return "Walk up early, exercise and take shower everyday.";
    }
    public static void main(String args[]) {
        System.out.print(new Year_2022().resolution());
    }
}
```

- We cannot declare a constructor of the abstract class.
- Compile time error.
- Output of this program is:
  Resolution:
  Walk up early, exercise and take shower everyday.

6) Consider the program given below and choose the correct option.

```
public interface Rectangle{
    abstract int areaRectangle(int length,int breadth);
}
public interface Circle{
    abstract int areaCircle(int radius);
}
public interface Shape extends Rectangle,Circle{

}
public class TwoDimension implements Shape{
    public int areaRectangle(int length,int breadth){
        return length*breadth;
    }
    public int areaCircle(int radius){
        return (int)(Math.PI*Math.pow(radius,2));
    }
}
public class Example{
    public static void main(String[] args){
        TwoDimension ref=new TwoDimension();
        System.out.println("The area of the rectangle is: "+ref.areaRectangle(5,2)
        +"\n"+"The area of the Circle is: "+ref.areaCircle(5));
    }
}
```

- The area of the rectangle is: 10 followe
- The area of the rectangle is: 10
The area of the Circle is: 78
- The area of the rectangle is: 10
The area of the Circle is: 75

7) Consider the Java program given below and predict the output.

```
public abstract class Base{
    public Base(){
        System.out.println("Base class constructor");
    }
}
public class Sub extends Base{
    public Sub(){
        System.out.println("Subclass constructor");
    }
}
public class Example {
    public static void main(String[] args){
        Base base=new Sub();
    }
}
```

- Compilation failed, you cannot define a constructor for abstract class.
- Compilation failed because of the statement below.
Base base=new Sub();
- Base class constructor
Subclass constructor

8) Consider the Java program given below and choose the correct option.

```
public interface City{
    public abstract void travel(String name);
}
public class Mumbai implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Mumbai");
    }
}
public class Hyderabad implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Hyderabad");
    }
}
public class Traveller{
    private City city;
    private String name;
    public Traveller(City city,String name) {
        this.city = city;
        this.name = name;
    }
    public void journey(){
        city.travel(name);
    }
}

public class Example{
    public static void main(String[] args) {
        Traveller traveller1=new Traveller(new Hyderabad(),"Johny");
        traveller1.journey();
        Traveller traveller2=new Traveller(new Mumbai(),"Virat");
        traveller2.journey();
    }
}
```

- Compilation failed because an interface cannot be a mem
- Compilation failed because Traveller constructor will not a
- Johny Travelling to Hyderabad
Virat Travelling to Mumbai
- null Travelling to null
null Travelling to null

9) Choose all the correct option(s) regarding the code given below.

```
public interface Academics{
    default void getName(){
        System.out.println("Academics : getName");
    }
    default void printName(){
        System.out.println("Academics : printName");
    }
}

public class University{
    private int univ_id;

    private class College implements Academics{
        private String college_name;
        public void getName(){
            System.out.println(college_name);
        }

        public void getID(){
            System.out.println(univ_id);
        }

        College(String name){
            college_name = name;
        }

        public College getReference(){
            return new College("IITMadras");
        }
    }
public class FClass{
    public static void main(String[] args) {
        University uni = new University();
        Academics acad = uni.getReference();
        acad.getName();
        acad.getID();
    }
}
```

- This code generates output:
IITMadras 0
- This code generates compilation error because default method getName is overridden.
- This code generates compilation error because method getReference does not return a valid College type object.
- This code generates compilation error because private instance variable univ id of outer class can not be directly accessed inside method getID of inner class.
- This code generates compilation error because getID method of class College is not accessible by acad object..

10) Consider the Java program given below.

```
import java.util.Date;
public class Timer{
    private Employee e;
    public Timer(Employee e){
        this.e=e;
    }
    @SuppressWarnings("deprecation")
    public void start(Date date){
        if(date.getDay()==6 || date.getDay()==0){
            e.notification1();
        }
        else{
            e.notification2();
        }
    }
}
public class Employee {
    public void check(){
        Timer obj=new Timer(this);
        Date d=new Date();
        obj.start(d);
    }
    public void notification1(){
        System.out.println("Happy weekend...");
    public void notification2(){
        System.out.println("Today is a working day, stay at work.");
    }
    public static void main(String[] args) {
        Employee e=new Employee();
        e.check();
    }
}
```

Note that the getDay() method returns an integer value from 0 to 6. Each integer represents a day of the week as follows:
0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday.
If this program in run on a Sunday, what will be the output?

- This code generates output: Today is a working day, stay at work.
- This code generates a compile time error.
- This code generates no output.
- This code generates the output:
Happy weekend...

Graded Assignment - 1

1) Consider the code given below.

```
public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public abstract class Collection{
    public void print() {
        System.out.println("no element");
    }
}

public class Queue extends Collection implements Printable{
    public void print() {
        super.print();
        System.out.println("print the queue");
    }
}

public class FClass{
    public static void main(String[] args) {
        Queue q = new Queue();
        q.print();
    }
}
```

What will be the output?

- no element
- not implemented
  print the queue
- not implemented
  print the queue
- no element
  print the queue
- It generates compiler error

2) Consider the code given below.

```
public interface Shape{
    public double area();
    public default double volume() {
        return -1.0;
    }
}

public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public class Rectangle implements Shape, Printable{
    private double w, h;
    public Rectangle(double w_, double h_) {
        w = w_;
        h = h_;
    }
    public double area() {
        return w * h;
    }
    public void print() {
        System.out.print(area() + " ");
        System.out.print(volume());
    }
}

public class FClass{
    public static void main(String[] args) {
        Shape s = new Rectangle(20.0, 50.0);
        s.print();
    }
}
```

What will be the output?

- 1000.0 followed by a runtime error
- not implemented
- 1000.0 -1.0
- It generates a compiler error

3)

```
public interface NumberType{
    public double norm();
    public static void print(double x) {
        System.out.println("norm is : " + x);
    }
}

public class ComplexNum implements NumberType{
    private int r, i;
    public ComplexNum(int r_, int i_) {
        r = r_;
        i = i_;
    }
    public double norm() {
        return Math.sqrt(r*r + i*i);
    }
}

public class FClass{
    public static void main(String[] args) {
        ComplexNum c = new ComplexNum(3, 4);
        //LINE1
    }
}
```

Identify the appropriate option(s) to fill in the blank at LIN

norm is : 5.0

- print(c.norm());
- c.print(c.norm());
- NumberType.print(c.norm());
- ComplexNum.print(c.norm());

4) Consider the code given below.

```
public interface DBIF{
    public void executeStatement(String qry);
}

public class Database{
    private ConnectionObj cobj = null;
    public DBIF connectDB(String u, String p) {
        if(isValidate(u, p))
            cobj = new ConnectionObj();
        return cobj;
    }
    public boolean isValidate(String u, String p) {
        //assume the vaidation is always true
        return true;
    }
    private class ConnectionObj implements DBIF{
        public void executeStatement(String qry) {
            System.out.println("Execute: " + qry);
        }
    }
}
public class FClass{
    public static void main(String[] args) {
        DBIF con = new Database().connectDB("test", "test");
        con.executeStatement("fetch students");
    }
}
```

Identify the appropriate option(s) to fill in blank at LINE1 such that the output becomes Execute: fetch students

- public DBIF connectDB(String u, String p)
- public ConnectionObj connectDB(String u, String p)
- public Database connectDB(String u, String p)
- public static ConnectionObj connectDB(String u, String p)

5) Consider the code given below.

```
public interface Inter{
    public abstract void greet();
}
public class Greetings{
    private String country;
    public void setCountry(String s){
        this.country = s;
    }
    public String getCountry(){
        return country;
    }
    public Inter checkCountry(){
        if(getCountry() == "India"){
            return new IndiaGreetings();
        }
        return new WorldGreetings();
    }
    private class IndiaGreetings implements Inter{
        public void greet(){
            System.out.println("Hello"+ " "+"India");
        }
    }
    private class WorldGreetings implements Inter{
        public void greet(){
            System.out.println("Hello"+ " "+"World");
        }
    }
}
public class Example {
    public static void main(String[] args) {
        Greetings g = new Greetings();
        g.setCountry("India");
        ---------Line 1---------
    }
}
```

Identify the appropriate option to fill in the blank at Line 1 to print Hello India

- g.greet();
- g.checkCountry().greet();
- g.IndiaGreetings.greet();
- g.Inter.IndiaGreetings.greet();

6) Consider the code given below.

```
public class Language{
    public void show(){
        System.out.println("In Language class");
    }
    public class Programming{
        public void show(){
            System.out.println("In Programming class");
        }
    }
}
public class Example {
    public static void main(String[] args) {
        ---------Line 1---------
    }
}
```

Identify the appropriate option to fill in the blank at Line 1 to print In Programming class

- Language.Programming.show();
- new Language().Programming.show();
- new Language().new Programming().show();
- Language.new Programming().show();

7)

```
public interface Animal{
    void sound();
    default void eat(String animal){
        System.out.println(animal+ " eats every day");
    }
}
public class Cat implements Animal{
    public void sound(){
        System.out.println("Cat meows");
    }
}
public class Dog implements Animal{
    public void sound(){
        System.out.println("Dog barks");
    }
}
public class Example{
    public static void main(String[] args){
        Animal oa1=new Cat();
        oa1.sound();
        oa1.eat("Cat");
        Animal oa2=new Dog();
        oa2.sound();
        oa2.eat("Dog");
    }
}
```

- Cat meows
  Dog barks
- Cat meows
  Cat eats every day
  Dog barks
  Dog eats every day
- Cat eats every day
  Dog eats every day

8)

```
public interface One{
    void print();
}
public interface Two extends One{
    void display();
}
public class Three implements Two{
    public void display(){
        System.out.println("This is display");
    }
}
public class Example{
    public static void main(String[] args){
        Three three=new Three();
        three.display();
    }
}
```

- Compilation fails because print() method has not been overridden in class Three.
- Compilation fails because one interface cannot extend another interface.
- Generates no output.
- This code generates the output: This is display

9) Choose all the correct option(s) regarding the code given bel

```
public class University{
    public int univ_id;

    private class College{
        private String college_name;

        public void getName(){
            System.out.println(college_name);
        }

        public College(String name){
        college_name = name;
        }
    }

    //As the class is private so we are using a public method
    //to return an object of the the inner class

    public College getReference(){
        return new College("IITMadras");
    }
}
public class FClass{
    public static void main(String[] args) {

University uni = new University();
(uni.getReference()).getName()
    }
}
```

getname() is func^n of College which is pvt. So to overcome this we need an abstract class or interface

- This code compiles successfully and generates output: IITMadras
- This code generates a compilation error because *college_name* is a private variable.
- This code generates a compilation error because *getReference()* is actually not returning a valid object of College type.
- This code generates a compilation error because the method *getName()* is not overridden and hence not accessible.

The Java code below models a certain functionality of a ropeway system, carrying tourists, that allows 6 travelers in each cabin. The travelers are let in one after the other into a cabin until the maximum capacity is reached (class CabinCounter). Then, the cabin is allowed to move on, and the next cabin is readied for boarding. A counter (class MasterCounter) is used to track the total number of travelers using the ropeway in a day. Based on the above information and the following code, answer questions **10** and **11**.

```
public interface Counter{
    public abstract void inform();
}
public class CabinCounter{
    private MasterCounter master;

    public CabinCounter(MasterCounter mc){
        master = mc;
    }
    public void performCount(){
        int currentCabinCount = 1;
        while (currentCabinCount <= 6) {
            Traveler t = new Traveler();
            if (t.hasValidTicket()){
                //Let the traveler get inside the current cabin
                currentCabinCount++;
            }
        }
        //let the cabin start moving
        ..... //Blank Line 1
    }
}
```

```
public class MasterCounter implements Counter{
    static int dayCount = 0;
    private CabinCounter cc;

    public MasterCounter(){
        cc = new CabinCounter(this); //A new cabin arrives
        cc.performCount();
    }
    public void inform(){
        this.incrementDailyCounter();
        cc = new CabinCounter(this); //A new cabin arrives
        ..... //Blank Line 2
    }
    public void incrementDailyCounter(){
        dayCount = dayCount + 6;
    }
    public static void main(String args[]) {
        MasterCounter masterCounter = new MasterCounter();
    }
}
```

```
public class Traveler{
    ....
    public boolean hasValidTicket(){
        //This method checks if the traveler has a valid ticket.
    }
}
```

11) Which should be the line at **Blank Line 2** so that the CabinCounter will start checking for the new cabin becoming full?

- There is no need to add any line because the constructor of CabinCounter will initiate the counting for the new cabin.
- cc.performCount();
- master.incrementDailyCounter();
- cc.inform();

10) Which should be the line at **Blank Line 1** so that the MasterCounter is notified of a cabin becoming full?

- mc.inform();
- master.inform();
- cc.incrementDailyCounter();
- MasterCounter cannot be notified because master is a private object inside CabinCounter.