BSCCS2005: Mock Quiz 1

1. Consider the Java code given below. [Dhannya:MCQ:2 points]

```
__________ FClass{              //Line 1
    abstract void fPrint();
}

__________ SClass{              //Line 2
    abstract void sPrint();
}

public class ChildClass extends FClass implements SClass{
    public void fPrint(){
        System.out.println("Inside fPrint");
    }
    public void sPrint(){
        System.out.println("Inside sPrint");
    }
}
public class MClass{
    public static void main(String []args){
        ChildClass p = new ChildClass();
        p.fPrint();
        p.sPrint();
    }
}
```

What should be filled in the given blanks so that the program generates the following output?

```
Inside fPrint
Inside sPrint
```

- ◯ `Line 1`: public interface, `Line 2`: public abstract class
- √ `Line 1`: public abstract class, `Line 2`: public interface
- ◯ `Line 1`: public abstract class, `Line 2`: public abstract class
- ◯ `Line 1`: public interface, `Line 2`: public interface

Solution: The blank in Line 1 cannot be an interface because ChildClass is extending FClass. So, options 1 and 4 are not correct. The blank in Line 2 cannot be an abstract class because ChildClass is implementing SClass. So, option 2 is the correct option. Observe that methods inside an interface can explicitly have the modifier `abstract` (even though it is implied).

2. Consider the Java code given below. [Dhannya:MCQ:2 points]

```
public class Engine{
    private class Fuel{
        String i_fuelType;
        public boolean isFossilFuel(String fuelType){
            if (fuelType == "Petrol" || fuelType == "Diesel")
                return true;
            else return false;
        }
    }
    public String maxMileage(String fuelType){
        Fuel f = new Fuel();
        if (_____________)                          //Line 1
            return "Low";
        else return "High";
    }
}
public class Example{
    public static void main(String[] args){
        Engine e = new Engine();
        int i = 0;
        String fType[] = {"Biodiesel","Diesel","Petrol","Steam"};
        for (i = 0; i<4; i++){
            System.out.println(_____________);     //Line 2
        }
    }
}
```

What should be filled in the blanks in `Line 1` and `Line 2`, respectively, such that the code prints the following output?

```
Low
High
High
Low
```

◯ `f.isFossilFuel(f.i_fuelType), e.maxMileage(fType[i])`

√ `!f.isFossilFuel(fuelType), e.maxMileage(fType[i])`

◯ `(f.i_fuelType == fuelType), e.isFossilFuel(fType[i])`

◯ The code generates a compilation error

Solution: The variable `i_fuelType`, which is the instance variable of the private class `Fuel` is not updated with the fuelType passed in from the main program. So, options 1 and 3 will not yield the correct answer. The code does not have any compilation error. Hence, option 2 is the correct answer. The public method maxMileage obtains the fuelType and invokes the isFossilFuel method inside the private class Fuel.

3. Consider the following Java code. [Dhannya:MCQ:2 points]

```
1  public class Base{
2      public Base(){
3          System.out.println("Inside Base");
4      }
5      public Base(String arg){
6          System.out.println("Inside Base "+arg);
7      }
8  }
9  public class Sub extends Base{
10     public Base(){
11         System.out.println("Inside Base of Sub");
12     }
13     public Sub(String arg){
14         System.out.println("Inside Sub "+arg);
15     }
16 }
17 public class BaseSubClass{
18     public static void main(String args[]){
19         Base b = new Sub("Test");
20     }
21 }
```

On removing which set of statements will this program generate the following output?

```
Inside Base
Inside Sub Test
```

- ◯ Lines 5 to 7
- √ Lines 10 to 12
- ◯ Lines 13 to 15
- ◯ Removing any set of lines cannot generate the desired output.

4. Consider the following Java code. [Dhannya:MCQ:2 points]

```
1  public abstract class AbstractOne{
2      abstract void myMethod();
3  }
4  public interface InterfaceOne{
5      default void myMethod(){
6          System.out.println("Inside Interface Default");
7      }
8  }
9  public class Child extends AbstractOne implements InterfaceOne{
10
11 }
12 public class TestAbstract{
13     public static void main(String args[]){
14         Child c = new Child();
15         c.myMethod();
16     }
17 }
```

What will be the output of this program?

- √ This code generates a compilation error because myMethod is not defined in class Child.
- ◯ This code generates a compilation error because an abstract method in one class cannot have a default implementation in another class.
- ◯ This code compiles correctly but generates a runtime error.
- ◯ This code generates the output: `Inside Interface Default`

5. Which of the following is not an advantage of static typing?

[Bhaskar:MCQ:2points]

- ◯ Compiler can make better code optimizations.
- ◯ Makes debugging of errors much easier.
- √ Makes the program less verbose.
- ◯ Reduces the number of runtime errors.

Solution: Static typing makes the program comparatively more verbose than dynamic typing because all type details has to be defined explicitly.

6. Consider the given code and choose the correct option regarding the same.

[Bhaskar:MCQ:2points]

```
public class Test{
    public static int x = 20;
    public static void fun(int arr[], int x){
        while(x > arr.length){
            x = x / 2;
        }
        for(int i=0;i<arr.length;i++){
            arr[i] = x;
        }
    }
    public static void main(String[] args) {
        int[] a = {2,3,5,7,11};
        int x = 100;

        fun(a,x);

        for(int i=0;i<a.length;i++){
            System.out.print(a[i]+ "  ");
        }
        System.out.println(x);
    }
}
```

- ◯ This code generates output:
  5 5 5 5 5 5

◯ This code generates output:
3 3 3 3 3 3

◯ This code generates output:
2 3 5 7 11 100

√ This code generates output:
3 3 3 3 3 100

◯ This code generates output:
5 5 5 5 5 100

Solution:

7. Consider the following code and choose all the correct option/s regarding the same.

[Bhaskar:MSQ:2points]

```
public interface Planet{
    default void show(){
        System.out.println("Planet Interface");
    }
}
public class Earth implements Planet{
    void show(){
        System.out.println("Earth Class");
    }
    void getPeriod(){
        System.out.println("365.26");
    }
}

public class Test{
    public static void main(String args[]){
        Planet p1 = new Earth();

        p1.show();
        p1.getPeriod();
    }
}
```

◯ This code generates the output:
Earth Class
365.26

◯ This code generates the output:
Planet Interface
365.26

√ This code generates compilation error because `p1.getPeriod()` can't be resolved.

√ This code generates compilation error because implementation of `show()` is not correct.

8. Consider the following code and choose all the correct option/s regarding the same.

[Bhaskar:MSQ:2points]

```
public class Account{
    String PAN;
    String IFSC;
    Account(Account obj){
        PAN = obj.PAN;
        IFSC = obj.IFSC;
    }
    public void getPAN(){
        System.out.println(PAN);
    }
}
public class SavingsAccount extends Account{
    double balance;
    SavingsAccount(SavingsAccount obj){
        balance = obj.balance;
    }
    SavingsAccount(String pan, double bal){
        balance = bal;
        PAN = pan;
    }
    public void getBalance(){
        System.out.println(balance);
    }
}
public class Test{
    public static void main(String[] args) {
        SavingsAccount acc1 = new SavingsAccount("CLX234",1024.22);
        Account acc2 = new SavingsAccount(acc1);
        acc2.getBalance();
        acc2.getPAN();
    }
}
```

◯ This code generates output:
1024.22
CLX234

◯ This code generates output:
0.0 null

◯ This code generates compilation error because `acc2.getPAN()` could not be resolved

√ This code generates compilation error because `acc2.getBalance()` could not be resolved

√ This code generates compilation error because the definition of appropriate constructor is missing.

9. Match the following terms with their appropriate description.

[ARUP:MCQ:2points]

A. Scope of a variable	1. Storage allocated is still on the stack
B. Lifetime of a variable	2. Requires manual memory management, programmer needs to explicitly request for allocation and deallocation of memory
C. Stack	3. Storage for local variables in functions
D. Heap	4. Run-time environment checks and cleans up dead storage
E. Automatic garbage collection	5. Variable in activation record at top of stack

◯ `A-1, B-4, C-3, D-2, E-1`

◯ `A-3, B-1, C-5, D-2, E-1`

√ `A-5, B-1, C-3, D-2, E-4`

◯ `A-2, B-5, C-2, D-3, E-4`

Solution:

10. What is the output of the following code?

[Nalini:MCQ:Medium: 3 points]

```
public class Pattern {
    public static void main(String[] args) {
        for(int i = 1; i <= 5; i++){
            for(int j = i; j < 5; j++){
                System.out.print(" ");
            }
            for(int j = 1; j <=5; j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

◯
```
    *
   **
  ***
 ****
*****
```

◯
```
*****
 *****
  *****
  *****
    *****
```

◯
```
  *
 ***
*****
 ***
  *
```

√
```
    *****
   *****
  *****
 *****
*****
```

11. Consider the following code and answer the question that follows.

[Dhannya:MCQ:Easy:2 points]

```
public class Student{
    private int rollnumber, chemMarks, phyMarks, mathsMarks;
    public int accessStudent(int c, int p, int m){
        _______________________________//Line 1
    }
    public void mutateStudent(int c, int p, int m) {
        _______________________________//Line 2
    }
}
public class QClass {
    public static void main(String[] args){
        Student s = new Student();
        int c = 50, p = 50, m = 50, t = 0;
        s.mutateStudent(c,p,m);
        t = s.accessStudent(c,p,m);
        System.out.println(t);
    }

}
```

What should be the statements in Line 1 and Line 2 if `accessStudent` and `mutateStudent` are respectively the accessor and the mutator methods for the class Student?

- ◯ Line 1: `return(chemMarks, phyMarks, mathsMarks);`
  Line 2: `chemMarks = c; phyMarks = p; mathsMarks = m;)`
- √ Line 1: `return(chemMarks+phyMarks+mathsMarks);`
  Line 2: `chemMarks = c; phyMarks = p; mathsMarks = m;)`
- ◯ Line 1: `chemMarks = c; phyMarks = p; mathsMarks = m;)`
  Line 2: `return(chemMarks+phyMarks+mathsMarks);`
- ◯ Line 1: `chemMarks = c; phyMarks = p; mathsMarks = m;)`
  Line 2: `return(chemMarks, phyMarks, mathsMarks);`

Solution: An accessor method in Java accesses the data in the instance variables, whereas a mutator method 'mutates' or changes the data in the instance variables. Further, since a method cannot return multiple values separated by commas, the solution follows.

12. Consider an educational institute which has a number of teachers. The teachers have schedules for their teaching days and times. Teachers can update their profiles and update their teaching subjects and teaching times. Students need to register the courses to audit them. Teachers teach the subject and provide various assignments to the students.

    Given the above scenario, identify the abstract types with the associated set of operations.

    [Nalini: MCQ: 2points]

    ◯
    - Teacher type with the operations – add and modify own profile, add and modify teaching timing, register for the course
    - Assignment type with the operation – add assignments
    - Student type

    ◯
    - Teacher type with the operations – add and modify own profile, add and modify teaching timing, add and modify teaching subject
    - Student type with the operation – register the courses
    - Assignment type

    ◯
    - Teacher type with the operations – add and modify own profile, add and modify teaching timing
    - Student type with the operation – register the courses
    - Assignment type

    √
    - Teacher type with the operations – add and modify own profile, add and modify teaching timing, add and modify teaching subjects, provide assignments
    - Student type with the operation – register the courses
    - Assignment type

Solution: The appropriate set abstract types associated with most proper related set of operations are:

- Teacher type with the operations – add and modify own profile, add and modify teaching timing, add and modify teaching subjects, provide assignments
- Student type with the operation – register the courses
- Assignment type

13. Identify the errors in the Java code given below. [Nalini: MSQ: 2points]

```
1 public abstract class Shape{
2
3 }
4 public class Rectangle{
5   private int length, breadth;
6   public Rectangle(int x, int y) {
7       length = x;
8       breadth = y;
9   }
10  public void area(){
11     System.out.println(length * breadth);
12  }
13 }
14 public class Square extends Rectangle, Shape{
15   private int side;
16   public Square(int x){
17     side = x;
18   }
19   public void area(){
20     System.out.println(side*side);
21   }
22 }
23 public class Example {
24   public static void main(String[] args) {
25    Rectangle r = new Rectangle();
26   }
27 }
```

- ◯ Line 11: illegal accessing of private variables
- ◯ Line 12: missing return statement
- √ Line 14: extending two classes is not allowed
- ◯ Line 20: illegal accessing of private variables
- ◯ Line 21: missing return statement
- ◯ Line 25: r cannot be created because methods are incomplete
- √ Line 25: r cannot be created because no matching constructor is found

14. Consider the following code. [ARUP: MCQ: 2points]

```
public class IntList extends List{
    private final int N = 5;
    private int iArr[];
    public IntList(int arr[]) {
        iArr = arr;
    }
    public void add(Object elem) {
        //add element elem to iArr
    }
    Traversal getTraversalObj() {
        return new Traversal();
    }
    private class Traversal implements BiTraversable{
        public void printForward() {
            for(int i = 0; i < N; i++)
                System.out.print(iArr[i] + " ");
        }
        public void printBackward() {
            for(int i = N - 1; i >= 0; i--)
                System.out.print(iArr[i] + " ");
        }
    }
}
public class FClass{
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40, 50};
        IntList iList = new IntList(arr);
        Traversable tr1 = iList.getTraversalObj();
        tr1.printForward();
        System.out.println();
        BiTraversable tr2 = iList.getTraversalObj();
        tr2.printBackward();
    }
}
```

Identify the appropriate definitions of `Traversable`, `BiTraversable` and `List`, such that the output is:

```
10 20 30 40 50
50 40 30 20 10
```

◯
```
public interface Traversable{
    void printForward();
```

```
}
public interface BiTraversable extends Traversable{
    void printBackward();
}
public interface List{
    public abstract void add(Object elem);
}
```

◯
```
public abstract class Traversable{
    public abstract void printForward();
}
public interface BiTraversable extends Traversable{
    void printBackward();
}
public abstract class List{
    public abstract void add(Object elem);
}
```

√
```
public interface Traversable{
    void printForward();
}
public interface BiTraversable extends Traversable{
    void printBackward();
}
public abstract class List{
    public abstract void add(Object elem);
}
```

◯
```
public interface Traversable{
    public abstract void printForward();
}
public interface BiTraversable implements Traversable{
    void printBackward();
}
public abstract class List{
    public abstract void add(Object elem);
}
```

15. Consider the following code and choose the correct option regarding the code.

[Bhaskar:MCQ:2points]

```
public class Restaurant{
    String[] arr;
    Restaurant(String[] a){
        arr = a;
    }

    public class Producer{
        String[] buffer;
        Consumer obj;
        Producer(){
            buffer = arr;
        }
        public void produce(){
            if(buffer[0] == null){
                System.out.println("Consumer has finished consuming");
                return;
            }
            buffer = new String[5];
            for(int i=0;i<buffer.length;i++){
                buffer[i] = "Item" + (i+1);
            }
            System.out.println("Producer has finished producing");
            new Consumer().consume(buffer);
        }
    }
    public class Consumer{
        public void consume(String[] buffer){
            for(int i=0;i<buffer.length;i++){
                buffer[i] = null;
            }
            new Producer().produce();
        }
    }
}
public class Test{
    public static void main(String args[]){
        String[] buffer = {"Item 0"};
        Restaurant obj = new Restaurant(buffer);
        Restaurant.Producer p = obj.new Producer();
        p.produce();
```

```
    }
}
```

◯ This code generates compilation error.

◯ This code generates the output:
Producer has finished producing
Consumer has finished consuming

◯ This code goes on printing the following pair of lines until the stack overflows
Producer has finished producing
Consumer has finished consuming

√ None of the above