# UTS PBO 2021/2022

## Ronde 1

1. Mengetahui dan menggunakan konsep dasar pada paradigma berorientasi objek tidak secara langsung membuat program menjadi lebih fleksibel , mudah dikembangkan dan dipelihara. Beberapa prinsip yang dikenal dengan OOP design principles perlu dipahami dan diterapkan oleh seorang programmer saat mendesain dan mengembangkan program berorientasi objek. Berikut ini adalah beberapa design principle yang dimaksud yaitu :

   A. Don't Repeat Yourself (Dry)
   B. Inheritance
   C. Open Closed Design
   D. Abstraction
   E. Programming to an interface
   F. Liskov Substitution
   G. Polymorphism
   H. Encapsulate What Varies
   I. Single Responsibility
   J. Interface Segregation
   K. Encapsulation
   L. Method Overriding
   M. Dependency Inversion Principle

2. Pada bagian //your code here, baris berikut ini tidak akan menyebabkan error...

```
interface Inter{}
class One{}
class Two extends One{}
class Three extends One{}
class Four extends Two implements Inter{}
class Five extends Four{}
class Six extends Three{}

public class Test{

    public static void main(String[] args) {
        //your code here
    }
}
```

   A. Inter inter = new Six();
   B. One one = new Five();
   C. Inter inter = new Four();
   D. Three three = new Four();

3. Modifier private mempunyai kontrol akses:

A. Class yang sama, package yang sama
B. Class yang sama
C. Class yang sama, package yang berbeda
D. Class yang berbeda, package yang berbeda

4. Output dari kode berikut adalah:

```
public class Handling {

    public static void main(String[] args) {
        try {
            System.out.println("Ujicoba Exception Handling");
            System.out.print("Hello" + " " + 1 / 0);
        } catch (ArithmeticException e) {
            System.out.print("World" + "\n");
        } finally {
            System.out.println("Ujicoba Selesai");
        }
    }
}
```

A. World\n
B. Ujicoba Exception Handling\n Hello\n Ujicoba Selesai\n
C. World\n Ujicoba Selesai\n
D. Ujicoba Exception Handling\n World\n Ujicoba Selesai\n

5. Berikut ini adalah pernyataan yang SALAH mengenai method overloading diantaranya :

Mensyaratkan proses inheritance sebelumnya dimana terdapat nama method yang sama pada child-class dan parent-class.
Memiliki dua atau lebih nama method yang sama dalam suatu class tetapi dengan parameter yang berbeda.
Suatu kondisi dimana terdapat sejumlah method dengan nama yang berbeda pada suatu class.
Merupakan salah satu cara mengimplementasikan konsep dasar OOP yaitu Polymorphism
Memiliki dua atau lebih nama method yang berbeda dari beberapa class dengan
parameter yang sama.

6. Konsep Pemrograman Berorientasi Objek berikut bermakna hanya menampilkan informasi yang penting bagi client... *

A. Encapsulation
B. Abstraction
C. Data Hiding
D. Data Binding

7. Nama lain Attributes dari sebuah objek adalah:

A. Methods

B. Properties
C. Classes
D. Functions
E. Abstractions

8. Output dari kode berikut ini jika dijalankan adalah

```
class One{
    public static void print(int i){
        System.out.println("Parent");
    }
}

class Two extends One{
    public static void print(byte b){
        System.out.println("Child");
    }
}

public class Test{
    public static void main(String args[]){
        One one = new Two();
        one.print(10);
    }
}
```

A. Parent
B. Child
C. Compilation error
D. Runtime error

9. Output dari kode berikut adalah:

```java
class A {

    public int i;
    private int j;
}

class B extends A {

    void display() {
        super.j = super.i + 1;
        System.out.println(super.i + " " + super.j);
    }
}

class Inheritance {

    public static void main(String args[]) {
        B obj = new B();
        obj.i = 1;
        obj.j = 2;
        obj.display();
    }
}
```

A. 2 2
B. 3 3
C. Compilation Error
D. Runtime Error

10. Berikut ini adalah pertanyaan yang BENAR

    Dalam sebuah class, method merepresentasikan state/status dari sebuah objek,
    dan variable merepresentasikan behaviour atau perilaku suatu objek.
    Memanggil method bisa digunakan untuk merubah nilai dari suatu variable pada
    objek.
    Status variabel di dalam objek-objek yang diinstansiasi dari class yang sama pasti akan memiliki nilai yang sama.
    Instance variable dengan non-access modifier “final” hanya bisa diinisialisasi nilainya satu kali saja.
    Anda harus menginisialisasi sebuah variable sebelum mendeklarasikannya.

11. Berikut identifier method pada Programming Java yang TIDAK sesuai dengan coding convention:

    run()
    draw()
    print_paper()

getName()
4action()
address()

12. Berdasarkan kode berikut, method “Car” mana yang akan dipanggil?

```
class Father {

    public void car() {
        System.out.println("Father's Car");
    }
}

class Son extends Father {

    public void car() {
        System.out.println("Son's Car");
    }
}

public class Sample {

    public static void main(String[] args) {

        Son john = new Son();
        john.car();
    }

}
```

A. Car pada kelas Father
B. Car pada kelas Son
C. Tidak ada yang dijalankan, karena terdapat ambiguitas antara Car pada kelas Father dan Son
D. Car pada kedua kelas, Father dan Son

13. Hubungan antara Kelas dan Objek yang benar adalah
A. Kelas merupakan cetakan untuk membuat Objek
B. Objek merupakan cetakan untuk membuat Kelas
C. Kelas diciptakan dari Objek
D. Objek dapat dibuat dari Kelas abstak

14. Manakah pernyataan berikut yang benar tentang blok Finally pada exception handling Java?
A. Blok Finally hanya dieksekusi jika exception dilemparkan ke blok Try

B. Blok Finally hanya dieksekusi jika exception dilemparkan ke blok Catch
C. Blok Finally hanya dieksekusi jika exception tidak dilemparkan ke blok Try atau Catch
D. Blok Finally dieksekusi terlepas dari apakah exception dilemparkan ke blok Try atau Catch

15. Berikut ini adalah pernyataan yang SALAH terkait penggunaan interface dan abstract class pada Java yaitu :

Pada interface dapat dideklarasikan variable yang secara implisit akan bersifat final dan static.
Pada abstract class dapat dideklarasikan variable yang secara implisit akan bersifat final dan static.
Interface dapat secara langsung diinstansiasikan menjadi sebuah objek.
Abstract class dapat secara langsung diinstansiasikan menjadi sebuah objek.
Satu class dapat mengimplementasikan lebih dari satu interface.
Satu class dapat diturunkan oleh lebih dari satu abstract class.
Interface dan abstract class bisa dijadikan sebagai tipe pada reference variable.
Interface hanya bisa memiliki method abstract sedangkan abstract class dapat
memiliki method abstract dan concrete.
Class abstract dapat mengimplementasikan interface, tetapi interface tidak dapat
mengimplementasi class abstract.

16. Konsep Pemrograman Berorientasi Objek berikut ini bermakna menyembunyikan kompleksitas.
a. Inheritance
b. Overloading
c. Data hiding
d. Abstraction

17. Berikut ini adalah implementasi encapsulations pada Java diantaranya :
- Membungkus variable dan method yang mengoperasikan data pada suatu entitas pada sebuah class.
- Membungkus satu atau lebih class yang berkaitan/sejenis dalam sebuah package.
- Membungkus package ke dalam sebuah module.
- Membungkus satu atau lebih class di dalam sebuah compilation unit.

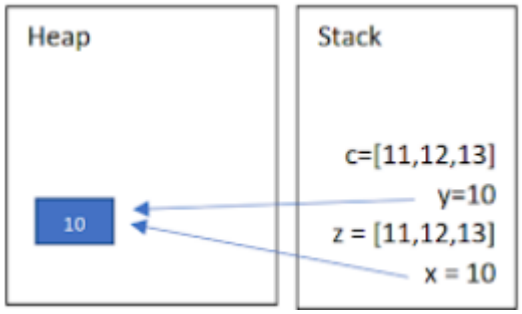
18. Berikut ini adalah fakta terkait garbage collections di Java yaitu :

- Garbage collection membuat java memory lebih efisien karena secara otomatis akan menghapus objek-objek yang sudah tidak memiliki referensi
- Garbage collector merupakan daemon yg melakukan garbage collection dan berada pada bagian Execution Engine pada Arsitektur JVM.
- Secara umum tahapan pada garbage collection adalah menandai objek-objek yang tidak memiliki referensi (marking) dan selanjutnya membersihkan objek-objek tersebut (sweeping).
- Membuat anonymous object, mengassign suatu variable reference yang sebelumnya mengarahkan ke sebuah objek menjadi null adalah beberapa kondisi yang memungkinkan garbage collector akan bekerja pada objek tersebut atau menjadikannya sebagai kandidat objek yang akan dibersihkan

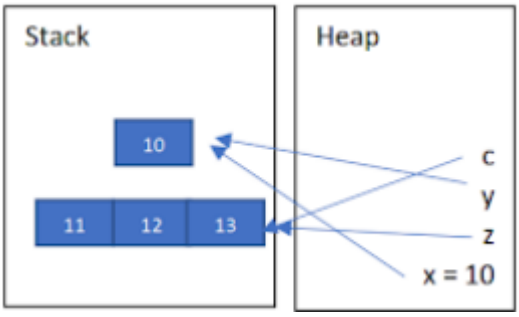
19. Perhatikan potongan kode berikut, model memory yang tepat untuk merepresentasikan potongan kode tersebut adalah ?

```
int x = 10;
int[] z = {11, 12, 13};
int y = x;
int[] c = z;
```

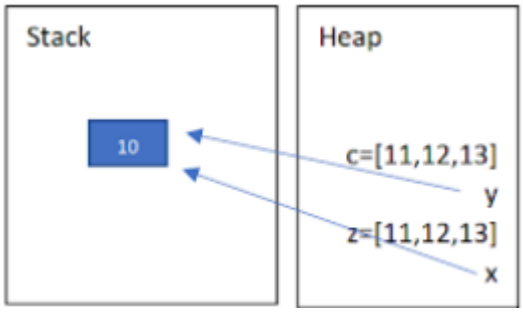
a.



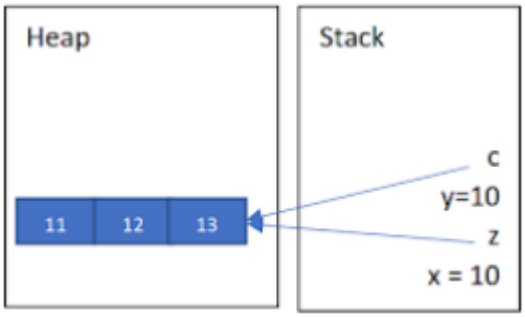

b.



c.




d.




20. Berikut kasus yang invalid saat melakukan overloading method:

a.
```
int add(int, int)
float add(int, int)
```

b.
```
add(int, int)
add(int, int, int)
```

c.
```
add(int, int)
add(int, float)
```

d.
```
add(int, float)
add(float, int)
```

21. Manakah pernyataan berikut yang TIDAK benar.
    a. Setiap kelas yang berisi abstract method harus dideklarasikan abstract.
    b. Kelas abstract hanya mendefinisikan struktur kelas bukan implementasinya
    c. Kelas abstract dapat diinisiasi
    d. Kelas abstract dapat di-inherit

22. Perhatikan compilation unit berikut ini! Method mana saja yang akan menyebabkan runtime error, logical error, dan compile error secara berurutan ?

```java
import java.util.*;
public class Errors {
    private double adderXY1(double x, double y) {
        double sum = x / y;
        return sum;
    }
    private integer subsXY(integer x, integer y) {
        integer subs = x - y;
        return subs;
    }
    private double countList(List<String> ls) {
        double count = ls.size() / 0;
        return count;
    }
    public static void main (String[] args) {
       Errors e = new Errors();
       double x = e.adderXY1(2, 4);
       List<String> ls = new ArrayList();
       ls.add("e1"); ls.add("e2");
       double l = e.countList(ls);
       integer i = subsXY(8,8);
    }
}
```

a. adderXY1, subsXY, countList
b. subsXY , adderXY1, countList
c. countList, adderXY1, subsXY
d. subsXY, countList, adderXY1
e. tidak ada error pada compilation unit diatas

23. Constructor digunakan untuk:
    a. Membuat user
    b. interface Membersihkan memori
    c. Menginstansiasi objek baru
    d. Membuat sub-class

24. Istilah yang tepat ketika subclass mendeklarasikan method yang memiliki tipe argument yang sama dengan method yang ada pada superclass
    a. Method override
    b. Method overload
    c. Operator overload
    d. Operator override
25. Method manakah pada kelas X yang tidak dapat di-override pada kelas Y?

```
class X {

    final public void m1() {

    }

    public void m2() {

    }

}
class Y extends X {

    // code for override
}
```

a. m1()
b. m2()
c. Baik m1() maupun m2() dapat di-override
d. Tidak satu pun m1() dan m2() dapat di-override

26. Output dari kode berikut adalah

```
public class Handling {

    public static void main(String[] args) {
        try {
            int i, sum;
            sum = 10;
            for (i = 1; i < 3; ++i) {
                sum = (sum / i);
            }
        } catch (ArithmeticException e) {
            System.out.print("0");
        }
        System.out.print(sum);
    }
}
```

a. 0
b. 5
c. Compilation Error
d. Runtime Error

27. Manakah yang BUKAN merupakan bagian dari konsep Pemrograman Berorientasi Objek
    a. Penyembunyian informasi
    b. Multitasking

c. Polymorphism
d. Inheritance

28. Berikut ini adalah pernyataan yang SALAH
    - Inheritance, digunakan untuk membuat suatu tipe entitas baru dari tipe entitas yang sudah ada sebelumnya, sehingga membentuk hubungan parent-child.
    - Java merupakan bahasa pemrograman yang mendukung konsep inheritance yang ada pada paradigma pemrograman procedural.
    - Multiple inheritance adalah sebuah fitur dimana sebuah class bisa menjadi turunan dari satu atau beberapa parent class. Pada java kita bisa menggunakan fitur tersebut dengan menggunakan keyword extends diikuti dengan nama-nama dari parent classnya.
    - Pada child class, variable atau method pada parent class yang berada di luar package akan secara otomatis diturunkan dan bisa diakses di dalam child class selama menggunakan access modifier protected atau public

29. Apakah tujuan utama dari Pemrograman Berorientasi Objek?
    a. Kemudahan pengembangan
    b. Monitoring pemeliharaan
    c. Enkapsulasi kode
    d. Penyembunyian informasi

30. Berikut ini adalah contoh identifier java yang diperbolehkan untuk menyimpan nilai bertipe int
    - why?
    - DID_THE
    - chicken
    - cro55
    - 5treet