

- 1) Pourquoi dans le langage Java on trouve toujours des types primitifs (non objet) comme le cas de **int**. Pourtant il y'a une classe **Integer** ?

Plus efficace et plus performant (0.5point)

- 2) Quel effet aura le mot clé final sur : un attribut, une méthode, ou une classe ?

Constant, no overwriding, no inheretence (1point)

- 3) Que veut dire le principe de la responsabilité unique dans la POO ?

Une classe doit avoir une cohésion et assure une seule tâche (0.5point)

- 4) Que signifie le concept du respect de contrat dans la POO en héritage ?

Ne change pas la spécification de la classe mère (1point)

Donner un exemple du non respect du contrat en héritage. (1point)

Class A{

public a ;

};

Class B extends A{

private a ;

}

- 5) Que signifie le type-casting dans la programmation en général ?

Changement du type à l'exécution (0.5point)

donner un exemple ? (1point)

float b=1.5 ;

int a= (int) b ;

- 6) Que signifie le down-casting ?

Changement du type de la classe mère vers la classe fille (0.5point)

donner un exemple (1point)

Class A{

};

Class B extends A{

}

A a=new B() ;

B b=(B) a;

- 7) Que signifie le up-casting ?

Changement du type de la classe fille vers la classe mère (0.5point)

Donner un exemple (0.5point)

Class A{

};

Class B extends A{

}

A a=new B() ;

- 8) A votre avis c'est quoi la différence entre type-casting (en général) et le up ou down casting ?

Up and down casting exige qu'il y a classe mère et classe fille. (0.5point)

- 9) A quoi sert le up-casting ? (2point)

- Supposant qu'on a une classe Parent A
- A a plusieurs classe filles: B, C, D, E, ...
- On a une certaine méthode *static SM* qui s'applique sur la classe A
- Je veux maintenant appliquer cette méthode sur des objets de B, C, D, E sans être obligé de la définir pour chaque classe??
- Solution: upcasting de B, C, D, E vers leur classe mère A

Montrer ça sur un simple exemple ? déjà vu au cours (2point)

Exemple:

```
public class Animal {
    String name;

    public void makeNoise() {
        System.out.println("I'm just an Animal");
    }
}
```

```
public class Dog extends Animal {
    @Override
    public void makeNoise() {
        System.out.println("Woof woof!");
    }

    public void growl() {
        System.out.println("Grrrrrr");
    }
}
```

```
public class Casting {
    public static void main(String[] args) {
        Animal myAnimal = new Dog();
        doAnimalStuff(myAnimal);
    }

    public static void doAnimalStuff(Animal animal) {
        animal.makeNoise();
    }
}
```

La méthode doAnimalStuff va être appliqué sur n'importe quel objet sous class de Animal → donc une seule méthode permet de tout faire

10) A quoi sert le down-casting ?

- essentiel lorsque, en fonction du Child, vous souhaitez accéder à des comportements de Child spécifiques. (1point)

Montrer ça sur un simple exemple ? vu dans le cours (2point)

Exemple



11) Soit le code java suivant :

- La classe Box est dite quoi ? **(1point)**
Une classe générique
- Proposer une définition du paramètre T. **(1point)**

```

public class Tool {
    private int size;
    private String description

    public Tool(int size, int description){
        this.size = size;
        this.description = description;
    }

    public String toString(){
        return "outil " + description + " de taille " + size;
    }
}

```

```

public class Box<T> {

    private T item;

    public void add(T item){
        this.item = item;
    }

    public T get() {
        return item;
    }

}

```

- Proposer un simple code qui utilise la classe Box. **(1point)**

```

public class ClasseClient {

    public static void main(String[] args) {
        Box<Integer> boite = new Box<>();
        boite.add(42);
        System.out.println(boite.get());

        Box<Tool> boiteOutils = new Box<>();
        boiteOutils.add(new Tool(5, "marteau"));
        System.out.println(boiteOutils.get());
    }
}

```

12) Soit le code java suivant :

```

public double calculerSalaires(ArrayList<? extends Employe> personnel) {
    double sommeFinale = 0;
    for (Employe e : personnel) {
        sommeFinale += e.getSalaireBrut();
    }
    return sommeFinale;
}

```

- Expliquer la signification du ? dans ce code. **(1point)**

Un joker

13) Soit le code suivant

```

public static void main(String[] args) {
    // TODO Auto-generated method stub
    // Insert code to start the application here.
    int i = 3;
    int j = 0;
    try {
        System.out.println("résultat = " + (i / j));
    } catch (Exception e) {
    } catch (ArithmeticException e) {
    }
}
}

```

- Pourquoi le compilateur signale une erreur ? **(1point)**
Une exception non atteignable car elle est déjà couverte par une exception générique
- Comment corriger l'erreur ? **(1point)**
Changer l'ordre des catch.