Comp 3021 Final exam – Spring 2021 – HKUST

Question 3: Abstract class and Interfaces [20 points]

a) (3 points)

Marker: Ling Hao

```
@Override                    // 1 point
public String howToCook() {
    // 2 points
    // each correct return value 1 point
    if (sauce){return "Cook with sauce added!";} // 1 point
    else{return "Cook without adding any sauce!";} // 1 point
}
```

b) (8 points)

Marker: Ling Hao

1. Must have @Override, otherwise lose that point
2. The input parameter of the method is StreetFood obj, but accept also Object obj, but then must cast ((StreetFood)obj) in the method, if this casting is not done, -1 point
3. default accessibility constraint of compareTo is public, if reduce this accessibility -1 point
4. Any other implementation that works correctly will get the full points

```
    @Override // 1 point
    // return type of method header 1 point, parameter 1 point,
    // if pass Object as parameter type and if cast correctly
    // below to StreetFood type in calling getPrice() and
    // getCookDate()then no mark deduction.
    public int compareTo(StreetFood sf){

            //1 point
        if (getPrice()>sf.getPrice())//direct use "price" is okay
            return 1;

            // 3 points, same price, compare the date
        else if (getPrice()==sf.getPrice()) {
            // must be sf.getCookDate(), because dateCooked is
            // private in the super class. If try to access
            // dateCooked directly -1 point

            if (getCookDate() > sf.getCookDate() {
                return 1;
            } else if (getCookDate == sf.getCookDate ())
                return 0;
            else
                return -1;
        }
        // 1 point, lower price
        else
            return -1;
    }
```

c) (9 points)

Marker: Ahmad

1. If consider "dateCooked" protected/ public -0.5 points each time, it is private in Food, StreetFood can't access it directly! For example if uses "dataCooked" directly to get it (but not the getter) then -0.5, and if assign it directly with a new Date object(but not through the setter), another -0.5. In total that will lose 0.5+0.5=1 point. Same also applies to "name" -0.5 points each time.
2. if there is the possibility of having CloneNotSupportedException, then it must be handled using try /catch. If "throws CloneNotSupportedException" -2 points
3. Any other implementation that works correctly will get the full points (including those who are using new and the constructors)
4. As we have discussed in the lectures, Cloneable is a marker interface, it has no abstract method inside. When a class "implements" it, it just means that Java will let the class doing the function call to clone(). clone() is a concrete method implemented in java.lang.Object, it will return a cloned object of the type java.lang.Object
4. Food does not implement clone() but it inherited the clone() method from java.lang.Object. so in StreetFood class, super.clone() will call the clone() in Object, and it will return a clone of the StreetFood as a java.lang.Object type object.

```
@Override // 1 point
//either return Object or StreetFood will be okay
// 1 point for protected/public accessibility
protected Object clone () {

// 1 point for declaring the reference of the cloned object
// outside the try{} block
// The declaration of the reference to the cloned object must
// not be in the try{} block, because if it is in the try{}
// block it will vanish once we leave the try{} block and we
// will not be able to return it
    StreetFood sf=null;

    try{
      //2 points for the call to clone() of super returns an
      // Object, need to cast properly, as by default it
      // returns an "Object"
      sf=(StreetFood) super.clone();
      //1 point
      sf.setName(new String(getName()));
      // 2 points, again need to cast properly
```

```
            //"dateCooked" and "name" are private and must be
            // accessed using 'getter' and 'setter'
            sf.setDateCreated((Date)getCookDate().clone());
        }
        catch(CloneNotSupportedException e){
        // this is optional and carries no point
        sf=null;
        }

        // 1 point
        // put the return to finally{} also okay
        // i.e. finally{ return sf;} okay
        return sf;

    }//end of clone()
```

One alternative answer:

```
@Override
protected Object clone () {
      StreetFood sf = new StreetFood(getName(), getServeHot(), sauce, price);
      String new_name = new String(getName());
      sf.setName(new_name);
      sf.setCookDate((Date)getCookDate().clone());
      return sf;
}
```

Question 4: GUI programming, anonymous inner classes and Lambda expressions [15 points]

Marker: Xu Hao

Answers:
TODO1:

```
// 0 point for this question if uses anonymous inner class instead of Lambda

                            // 1 point for correct lambda syntax
                            // 1 point for KeyEvent at the left of ->
                            // deduct 4 points if syntactically correct, but
                            // does not use the correct JavaFX methods.
        pane.setOnKeyPressed((e)->{ // 4 points for UP, 4 points for DOWN, if
                                    // if mistaken "dot" with "circle" below,
                                    // deduct 1 point for once only.
                        switch (e.getCode()) {
                            case UP: {
                                Line newLine = new Line(dot.getCenterX(),
dot.getCenterY(), dot.getCenterX() , dot.getCenterY()- movementDelta);
                                dot.setCenterY(dot.getCenterY() - movementDelta);
                                pane.getChildren().add(newLine);
                                break;
                            }
                            case DOWN: {
                                Line newLine = new Line(dot.getCenterX(),
dot.getCenterY(), dot.getCenterX() , dot.getCenterY()+ movementDelta);
                                dot.setCenterY(dot.getCenterY() + movementDelta);
                                pane.getChildren().add(newLine);
                                break;
                            }

                        }
                    });
```

Marker: Peisen

TODO2:

```
// 0 point for this question if uses Lambda instead of anonymous inner class
// deduct 2 points if syntactically correct, but
// does not use the correct JavaFX methods.

pane.setOnMouseClicked(
// 1 point for "new"
// 1 point for "EventHandler<MouseEvent>()"
// 3 points for the implementation
new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent e) {
                dot.setCenterX(e.getSceneX());//e.getX() also okay
                dot.setCenterY(e.getSceneY());//e.getY() also okay
            }
        }
);
```

Question 5: Multi-threading, synchronization and condition object [10 points]

Marker: Yixin
Answer:

```
// TODO 1:
// 2 points
lock.lock(); // Acquire the lock

// TODO 2:
// (6 points, 1 point for the while loop, 2 points for the checking the
// condition in the while loop, 3 points for calling
// printCondition.await(); correctly
// any other approach that works will also be okay

while ((charPrinted)%3!=(threadChar-65)%3){
        printCondition.await();
}


// TODO 3:
// 2 points
printCondition.signalAll();
```

.

~~~~~~WISH YOU ALL A HAPPY AND FRUITFUL SUMMER~~~~~~
~~~~~~