Programming II (QUIZ, Design Patterns)

Design Patterns → Creational (of object)
↙ ↓ ↘
Singleton Factory etc

In FN → we only need 1 player in the game time so we can use singleton

* we make private constructor so new ("btedrab")
ex: Player p = ~~new~~ Player();

* we call get instance to get the only instance (player) in the game.

* That was Singleton

Let's look at Factory Method Design

* we create Factory of fruits
as Fruit → Strawberry
↳ Melon

(Casting Circle c = (Circle)s;
to print radius for ex)

! Note ! Concrete Class (not abstract)

* we can make singleton factory & (or) we have one factory in "FN" using Factory of factories for example.

- Abstract factory

* We have → Bombs
→ Fruits } Factories

Fruit Factory
BombF. — } → extend AbstractFactory

we create Fruit return null in Bomb & viceversa because of the AbstractFactory

We create Factory of factories
AbstractFactory (abstract) → FruitFactory (singleton)
→ BombFactory (singleton)

& we can make it Interface

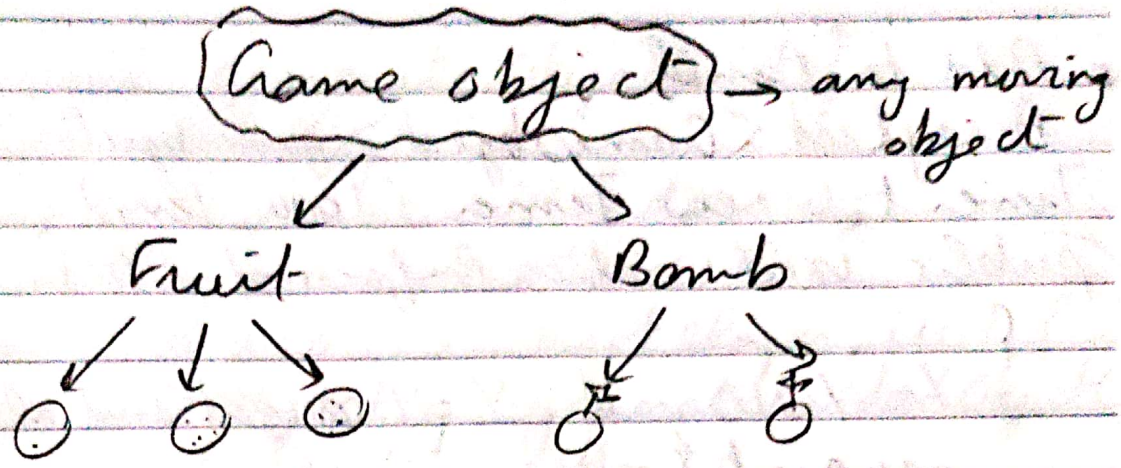
* We collected 3 Design patterns, remaining (2)

* AbstractFactory contains abstract methods.

* Interfaces in the pdf are good

Take a look at the project.

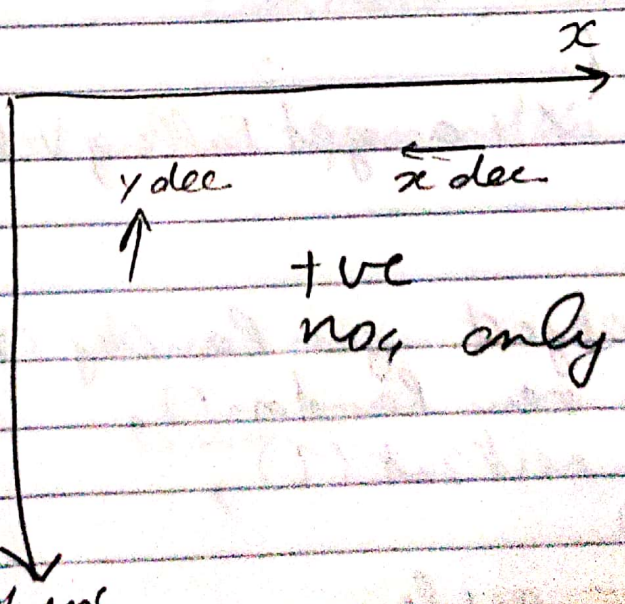
No:---------- Date:----------



* We have GamePanel Class that extends JPanel (Drag from Projects)

```
protected void paintComponent(Graphics g)
{
    super.paintComponents(g);
    g.draw___();
    g.drawImage(a.getB__I().getImg(), 0, 0, this)
}
```

```
Apple a = new Apple();
```



No: ..... Date: .....

* Add to Panel

```
ArrayList<GameObject> gameObjects
Timer t = new Timer(1000, this); , init y,l;
public void actionPerformed(ActE e)
  {
    Y = Y - 10 : // Y = Y - a.getInitialVelocity();
    repaint(); }
                ↳ "bana"
public GamePanel() {
  t.start
}

public void actionPerformed(ActionEvent e)
{
  if (a.isMovingUp()) {
      Y = Y - a.getInitialVelocity();
      if (Y<20) {
          a.setMovingUp(false);
      }
  } else {
      Y = Y + a.getFallingVelocity();
  }

public GameObject genRanObj() {
 Random r = new Random();
 int x = r.nextInt(2);
 if (x==0)
     return apple
 else
     return orange;
```

```
public void generateObjects() {
    int noofShapes;
    Random r = new Random();
    noofShapes = r.nextInt()+1;
    for (int i=0, i<noofShapes; i++) {
        GameObject g = generateRandomObject();
        g.setXlocation(xloc);
        g.setYlocation(yloc);
        gameobjects.add(g);
        xloc += 80; // To separate objects
    }
}
```

Add to Action performed:

```
if (newX > 80)
  { go.setMovedOff(true); }
```

→ Next lecture slicing
Command, Memento,

File to save game (parameters needed)