**Lab2-Greenfoot and Processing**
**Gumball Machine**

## Encapsulation:-

A process of wrapping code and data together into a single unit. We can use setter and getter methods to set and get the data in it.

The 'displayMessage(String msg)' method, when called, invokes the setText() method with its own data depending on which method called it. Control over data is maintained.

**Alien.java**

```
public void displayMessage(String msg){
…..
 m.setText(msg);
….
}
```

**Message.java**

```
public void setText( String msg)
   {
      GumballWorld g = (GumballWorld) getWorld();
      image.clear();
      image.setColor( java.awt.Color.getHSBColor(169, 100, 68) ) ;
      image.fill() ;
      image.setColor( java.awt.Color.BLACK ) ;
      image.setFont(new java.awt.Font("Arial", java.awt.Font.PLAIN, 18));
      image.drawString( msg, 250, 25 );

   }
```

**RandomPicker.java**

```
public void pickGumball(){
….
 displayMessage("Gotcha Gotcha!”);
….
}
```

**GreenPicker.java**

```
 public void pickGumball(){
….
 displayMessage("Go green”);
…..
}
```

## Inheritance

A mechanism in which a child acquires all the properties and behaviors of parent object.

Penny and Quarter are categorized under a coin and are capable of acquiring the parents properties.
FakeQuarter is inherited from Quarter which in-turn is a child to the coin class.

**Coin.java**

```
public void act()
   {
      int jerry, nibbles ;

      if(Greenfoot.mouseDragged(this)) {
         MouseInfo mouse = Greenfoot.getMouseInfo();
         jerry=mouse.getX();
         nibbles=mouse.getY();
         setLocation(jerry, nibbles);
      }
   }
```

RedGumball, BlueGumball, GreenGumball come under the Gumball class and inherit the act functionalities from the parent.

**BlueGumball.java:**

```
public void act()

   {
      if(Greenfoot.mousePressed(this)) {
         World world = getWorld();
         world.removeObject( this ) ;

      }
```

**RedGumball.java**

```
public void act()
   {
      if(Greenfoot.mousePressed(this)) {
         World world = getWorld();
         world.removeObject( this ) ;

      }
```

**Gumball.java**

```
 public Gumball()
   {
      GreenfootImage image = getImage() ;
      image.scale( 50, 50 ) ;
   }

   public void act()
   {
      if(Greenfoot.mousePressed(this)) {
         World world = getWorld();
         world.removeObject( this ) ; }
```

Picker and Inspector come under the Alien class which inherit properties of alien which enables them to decide and send the gumball to the respective alien.
RandomPicker and GreenPicker come under the Picker class.

**Method Overloading:-**

**Alien.java:**

```
public void displayMessage(String msg){

        Message m = new Message();
        m.setText(msg);
        GumballWorld g = (GumballWorld) getWorld();
        g.addObject(m,120,550);
        Greenfoot.delay(80);
        g.removeObject(m);
    }
```

**Inspector.java:**

```
public class Inspector extends Alien{
public void displayMessage(String msg, int x, int y){

        Message m = new Message();
        m.setText(msg);
        GumballWorld g = (GumballWorld) getWorld();
        g.addObject(m,120,550);
        Greenfoot.delay(100);
        g.removeObject(m);
    }
}
```

## Alien.java

```
 public void displayMessage(String msg){

      Message m = new Message();
      m.setText(msg);
      GumballWorld g = (GumballWorld) getWorld();
      g.addObject(m,120,550);
      Greenfoot.delay(80);
      g.removeObject(m);
}
```

The setText()method is invoked for every displayMessage() called by different classes. The message to be displayed is added to the world with predefined coordinates.
After a small time lapse the message disappears from the screen.

## RandomPicker.java

```
public void pickGumball(){
      Message m1 = new Message();
      Message m2 = new Message();
      GumballWorld g = (GumballWorld) getWorld();
      int gumType = Greenfoot.getRandomNumber(g.getgumballMap().size());
      Gumball gumball = (Gumball) g.getgumballMap().get(gumType);
      g.addObject(gumball, 400, 350);
      moveToPicker(gumball);
      displayMessage("Gotcha Gotcha!");
      g.removeObject(gumball);
  }
public void moveToPicker(Gumball b){

      b.turnTowards(648, 94);
      for(int i=0;i<75;i++){
         b.move(5);
         Greenfoot.delay(1);
      }
```

The random picker is designed to pick randomly and return a gumball. Every randomly picked gumball is added to the world and displays a message depending on the alien who receives it. The gumboil then disappears from the screen.

## Message.java

This class has been created to handle message displays on the screen.

```
  public void setText( String msg)
  {
     GumballWorld g = (GumballWorld) getWorld();
     image.clear();
     image.setColor( java.awt.Color.getHSBColor(169, 100, 68) ) ;
     image.fill() ;
     image.setColor( java.awt.Color.BLACK ) ;
     image.setFont(new java.awt.Font("Arial", java.awt.Font.PLAIN, 18));
     image.drawString( msg, 250, 25 );

  }

  public void act()
  {
     if(Greenfoot.mousePressed(this)) {
        World world = getWorld();
        world.removeObject( this ) ;

     }
```

The purpose of this class is to handle setText() calls from different displayMessage() methods. The font size, type and colors have been specified.

The drawString() draws the text given by the specified string, using the current font and color.

Reason for using 3.0.4:-
Greenfoot 3.0.4 supports java.awt while 3.1.0 has come up with its own classes(no examples).

### Inspector.java

```
public void inspectCoin(Coin c){
      if(c.getClass() == Quarter.class){
         pickPicker();
         GumballWorld gw = (GumballWorld) getWorld();
         gw.addObject( new Quarter(), 68, 156 ) ;
      }

      if(c.getClass() == FakeQuarter.class){
         GumballWorld gw = (GumballWorld) getWorld();
         gw.addObject( new FakeQuarter(), 66, 258 ) ;
         moveInspector();
         displayMessage("Fake quarter alert!", 400, 400);
      }

      if(c.getClass() == Penny.class){
         GumballWorld gw = (GumballWorld) getWorld();
         gw.addObject( new Penny(), 57, 71 ) ;
         moveInspector();
         displayMessage("Thats just a penny. We need a quarter!", 400, 400);
      }

   }
```

Inspects which coin is entered into the slot using getClass(). If any condition is satisfied the corresponding coin is added into the slot.

```
public void pickPicker(){
      GumballWorld g = (GumballWorld) getWorld();i
int type=Greenfoot.getRandomNumber(g.getpickerMap().size());
      Picker picker = (Picker) g.getpickerMap().get(type);
      picker.pickGumball();


   }
```

The pickpicker() selects a picker alien randomly from the getpickerMap() hash. The type is then rounded to its corresponding picker and the respective gumball is selected.

### GreenPicker.java

```
public void pickGumball(){
      Message m = new Message();
      Message m2 = new Message();
      GumballWorld g = (GumballWorld) getWorld();
      Gumball gumball = (Gumball) g.getgumballMap().get(2);
      g.addObject(gumball, 400, 350);
      moveToPicker(gumball);
      displayMessage("Go green");
      g.removeObject(gumball);

   }
public void moveToPicker(Gumball blob){

      blob.turnTowards(680, 441);
      for(int i=0;i<42;i++){
         blob.move(5);
         Greenfoot.delay(1);
      }
```

The greenpicker always selects the green gumball

**getgumballMap().get(2) -** This always selects the green gumball(as entered in the hash). The gumball is then displayed on the screen with a message and then removed after a small delay.

**Coin.java**

```
if(Greenfoot.mouseDragged(this)) {
        MouseInfo mouse = Greenfoot.getMouseInfo();
        jerry=mouse.getX();
        nibbles=mouse.getY();
        setLocation(jerry, nibbles);
    }
```

To return the current X and Y positions of the mouse

**GumballMachine.java**

```
public void act()
  {
    if(coinPresent){
        if(Greenfoot.mouseClicked(this)){
            displayMessage("Inspecting coin...",100,700);
            moveGumball();
            inspector.inspectCoin(savedCoin);
            coinPresent = false;
            coin = null;
            coinPresent = false;
        }
    }
        else if(Greenfoot.mouseClicked(this)){
            displayMessage("Please enter a coin!",150,600);

        }

    coin = (Coin)getOneObjectAtOffset( +10, +10, Coin.class);
    if(coin != null){

        if(!coinPresent){
            coin = (Coin) coin;
            savedCoin = coin;
            removeCoin(coin);
            waitingForCrank = true;
            coinPresent = true;
            displayMessage("Turn the crank!",150,600);
        }

    }

  }
```

Checks whether a coin is inserted into the machine and displays a message accordingly.

**GumballWorld.java**

```
 HashMap<Integer, Picker> pickerHash = new HashMap();
 HashMap<Integer, Gumball> gumballHash = new HashMap();
private void prepare()
   {
       pickerHash.put(1,greenpicker);

      gumballHash.put(0,new RedGumball());
      gumballHash.put(1,new BlueGumball());
      gumballHash.put(2,new GreenGumball());

      gumballmachine.setinspector(inspector);


   }

   public HashMap getpickerMap(){
      return pickerHash;
   }

   public HashMap getgumballMap(){
      return gumballHash;
   }
```

A hash map with all types of gumballs is created and is used by different pickers to deliver the corresponding colored gumball.