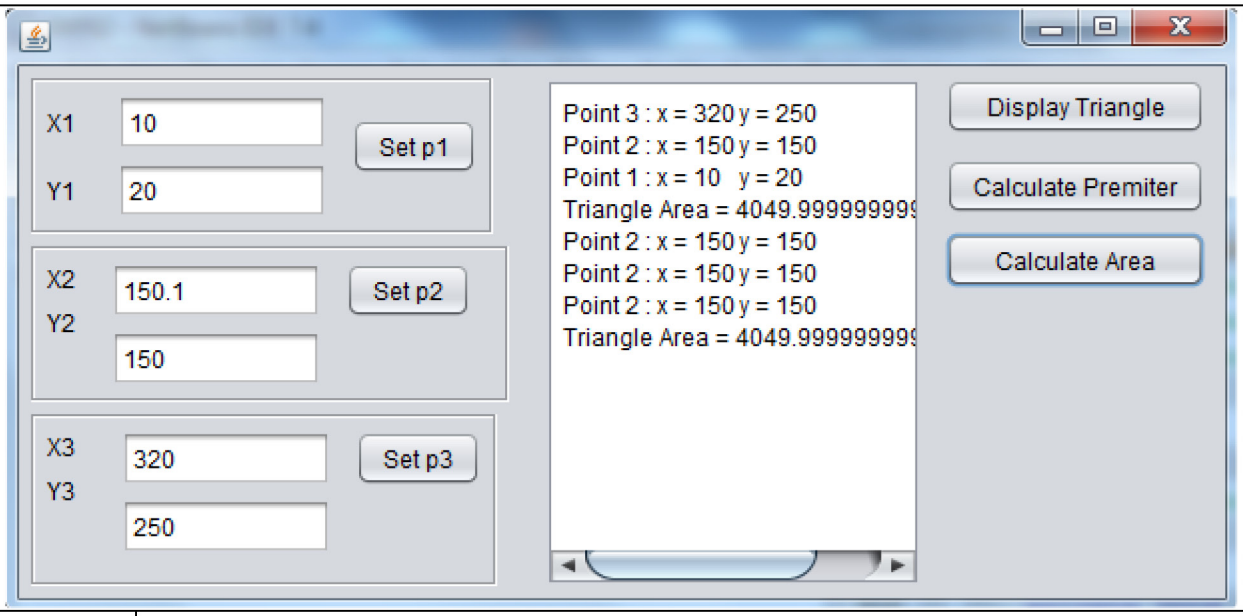
Fundamentals of Programming II Lab 13

EXP02



setP1 button	try {
	t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
	} catch (Exception e) {
	
	}
	jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

setP2 button	try {
	t.setP2(Integer.parseInt(jTextField3.getText()), Integer.parseInt(jTextField4.getText()));
	} catch (Exception e) {
	
	}
	jTextArea1.append("Point 2 : x = " + t.getP2().getX() + "\ty = " + t.getP2().getY() + "\n");

setP3 button	try {
	t.setP3(Integer.parseInt(jTextField5.getText()), Integer.parseInt(jTextField6.getText()));
	} catch (Exception e) {
	
	}
	jTextArea1.append("Point 3 : x = " + t.getP3().getX() + "\ty = "

		+ t.getP3().getY() + "\n");

EXP03	SetP1	try {
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

EXP04	SetP1	try {
		if(Integer.parseInt(jTextField1.getText())<0)
		throw new Exception("Positive Number Required : "+jTextField1.getText());
		if(Integer.parseInt(jTextField2.getText())<0)
		throw new Exception("Positive Number Required : "+jTextField2.getText());
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

EXP05	SetP1	try {
		if(Integer.parseInt(jTextField1.getText())<0)
		throw new Exception("Positive Number Required : "+jTextField1.getText());
		if(Integer.parseInt(jTextField2.getText())<0)
		throw new Exception("Positive Number Required :

		"+jTextField2.getText());
		if (Integer.parseInt(jTextField1.getText()) > jPanel4.preferredSize().width) {
		throw new Exception("X is out of boundries, should be < " + jPanel4.preferredSize().width);
		}
		if (Integer.parseInt(jTextField2.getText()) > jPanel4.preferredSize().height) {
		throw new Exception("X is out of boundries, should be < " + jPanel4.preferredSize().height);
		}
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

EXP06	SetP1	try {
		if(Integer.parseInt(jTextField1.getText())<0)
		throw new SecurityException("Positive Number Required : "+jTextField1.getText());
		if(Integer.parseInt(jTextField2.getText())<0)
		throw new SecurityException("Positive Number Required : "+jTextField2.getText());
		if (Integer.parseInt(jTextField1.getText()) > jPanel4.preferredSize().width) {
		throw new SecurityException("X is out of boundries, should be < " + jPanel4.preferredSize().width);
		}
		if (Integer.parseInt(jTextField2.getText()) > jPanel4.preferredSize().height) {
		throw new SecurityException("X is out of boundries, should be < " + jPanel4.preferredSize().height);

```
            }
            t.setP1(Integer.parseInt(jTextField1.getText()),
Integer.parseInt(jTextField2.getText()));
        }catch(SecurityException e){
            JOptionPane.showMessageDialog(this, "Out of Bound
Error\n"+e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "illegal Input
Error\n"+e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = "
+ t.getP1().getY() + "\n");
```

EXP07

SetP1

```
        try {
            try {
                if (Integer.parseInt(jTextField1.getText()) < 0) {
                    throw new Exception("Positive Number Required : " +
jTextField1.getText());
                }
                if (Integer.parseInt(jTextField2.getText()) < 0) {
                    throw new Exception("Positive Number Required : " +
jTextField2.getText());
                }
            } catch (Exception e) {
                JOptionPane.showMessageDialog(this, e.getMessage(),
"Error", JOptionPane.ERROR_MESSAGE);
            }
            try {
                if (Integer.parseInt(jTextField1.getText()) >
jPanel4.preferredSize().width) {
                    throw new Exception("X is out of boundries, should be
< " + jPanel4.preferredSize().width);
                }
                if (Integer.parseInt(jTextField2.getText()) >
jPanel4.preferredSize().height) {
                    throw new Exception("X is out of boundries, should be
```

		< " + jPanel4.preferredSize().height);
		}
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

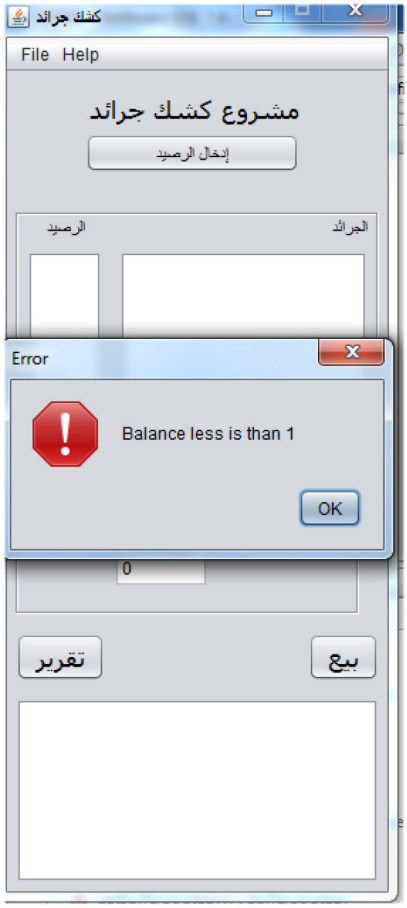
EXP08	CPointXY	void setX(int X) throws Exception
		{
		if(X<0) throw new Exception("Positive number required");
		this.X=X;
		}
		void setY(int Y) throws Exception
		{
		if(Y<0) throw new Exception("Positive number required");
		this.Y=Y;
		}

	CTeriangle	void setP1(int x, int y) throws Exception{
		p1.setX(x);
		p1.setY(y);
		}
		void setP2(int x, int y) throws Exception{
		p2.setX(x);
		p2.setY(y);
		}
		void setP3(int x, int y) throws Exception{
		p3.setX(x);
		p3.setY(y);

		}

	Form
	
	SetP1
		try {
		if (Integer.parseInt(jTextField1.getText()) > jPanel4.preferredSize().width) {
		throw new Exception("X is out of boundries, should be < " + jPanel4.preferredSize().width);
		}
		if (Integer.parseInt(jTextField2.getText()) > jPanel4.preferredSize().height) {
		throw new Exception("X is out of boundries, should be < " + jPanel4.preferredSize().height);
		}
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " + t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

EXP09



NewsPaper

```
void initNewsPaper(String name, int balance,
double netPrice,
        double sellPrice) throws Exception {
    if (balance < 1) {
        throw new Exception("Balance less is than
1");
    }
    if (netPrice < 0) {
        throw new Exception("Net Price is less than
0");
    }
    if (sellPrice <= netPrice) {
        throw new Exception("Sell price is equals or
less than net price");
    }
    this.name = name;
    this.balance = balance;
    this.netPrice = netPrice;
```

		this.sellPrice = sellPrice;
		}
		void sellNewsPaper() throws Exception {
		if (balance == 0) {
		throw new Exception("zero balance can't sell selected utem");
		}
		balance--;
		}

	NewsPaperBooth	void addNewsPaper(String name, int balance, double netPrice,double sellPrice) throws Exception {
		np[items].initNewsPaper(name, balance, netPrice, sellPrice);
		items++;
		}
		void sellNewsPeper(int item) throws Exception
		{
		np[item].sellNewsPaper();
		costInHand+=np[item].getNetPrice();
		sellInHand+=np[item].getSellPrice();
		profit=sellInHand-costInHand;
		}

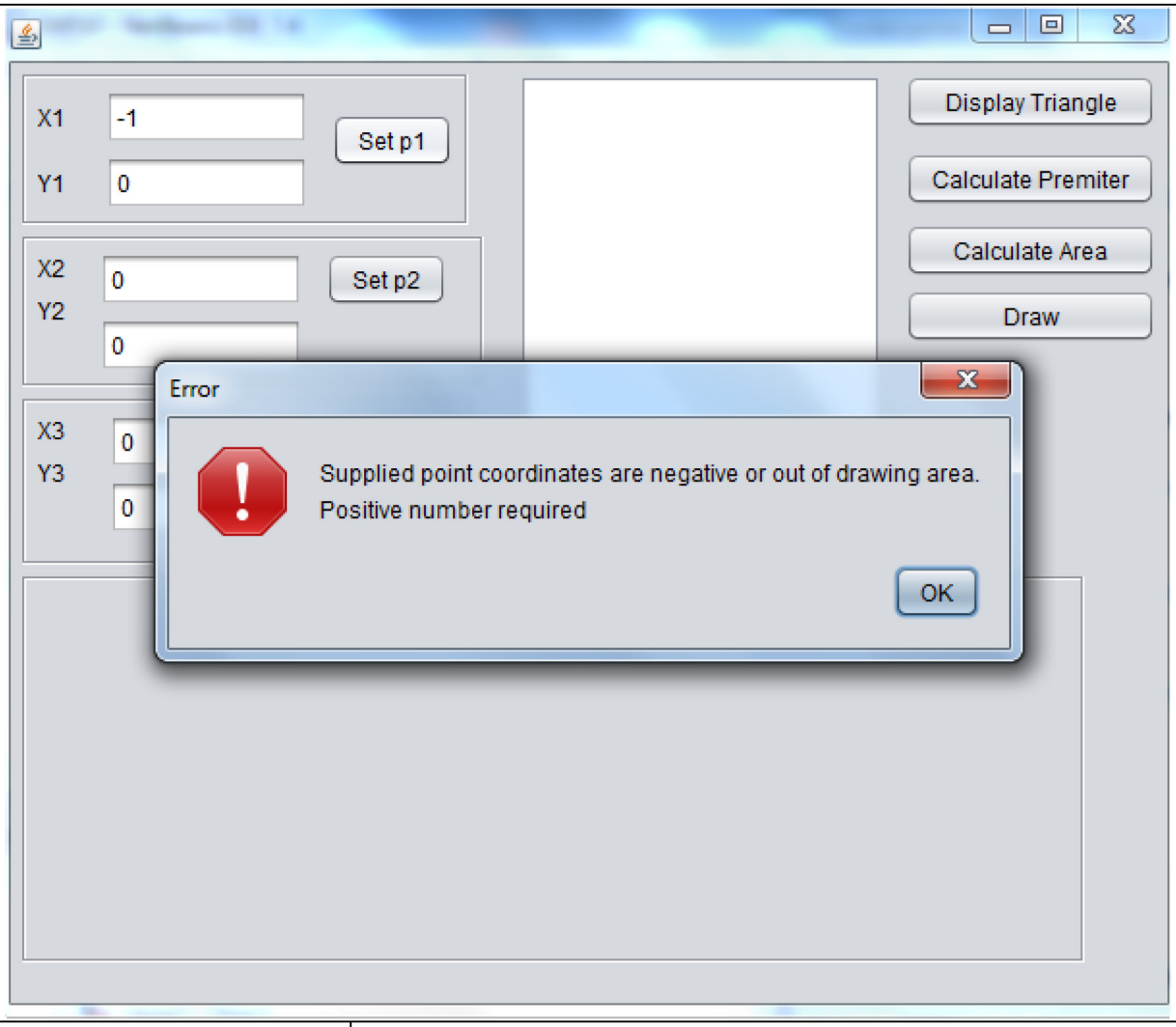
	form	

	formWindowActivated	if (snpf.press.equalsIgnoreCase("apply")) {
		this.setEnabled(true);
		try {
		npb.addNewsPaper(snpf.name,
		Integer.parseInt(snpf.balance),
		Double.parseDouble(snpf.netPrice),
		Double.parseDouble(snpf.sellPrice));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}

		DefaultListModel dlm1 = new DefaultListModel();
		DefaultListModel dlm2 = new DefaultListModel();
		for (int n = 0; n < npb.items; n++) {
		dlm1.addElement(npb.np[n].getName());
		}
		for (int n = 0; n < npb.items; n++) {
		
		dlm2.addElement(npb.np[n].getBalance());
		}
		
		jList1.setModel(dlm1);
		jList2.setModel(dlm2);
		} else if (snpf.press.equalsIgnoreCase("cancel")) {
		this.setEnabled(true);
		} else if (snpf.press.equalsIgnoreCase("1")) {
		snpf.setVisible(true);
		}
		snpf.press = "0";

	بيع	try {
		int[] selected = new int[totalItems];
		selected = jList1.getSelectedIndices();
		for (int n = 0; n < selected.length; n++) {
		npb.sellNewsPeper(selected[n]);
		}
		update();
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}

EXP10



BoundriesException

```
public class BoundriesException extends Exception{

   BoundriesException()
   {
      super();
   }
   BoundriesException(String msg)
   {
      super(msg);
   }
   @Override
   public String getMessage()
   {
      return "Supplied point coordinates are negative
or out of drawing area."
            +"\n"+super.getMessage();
```

		} }

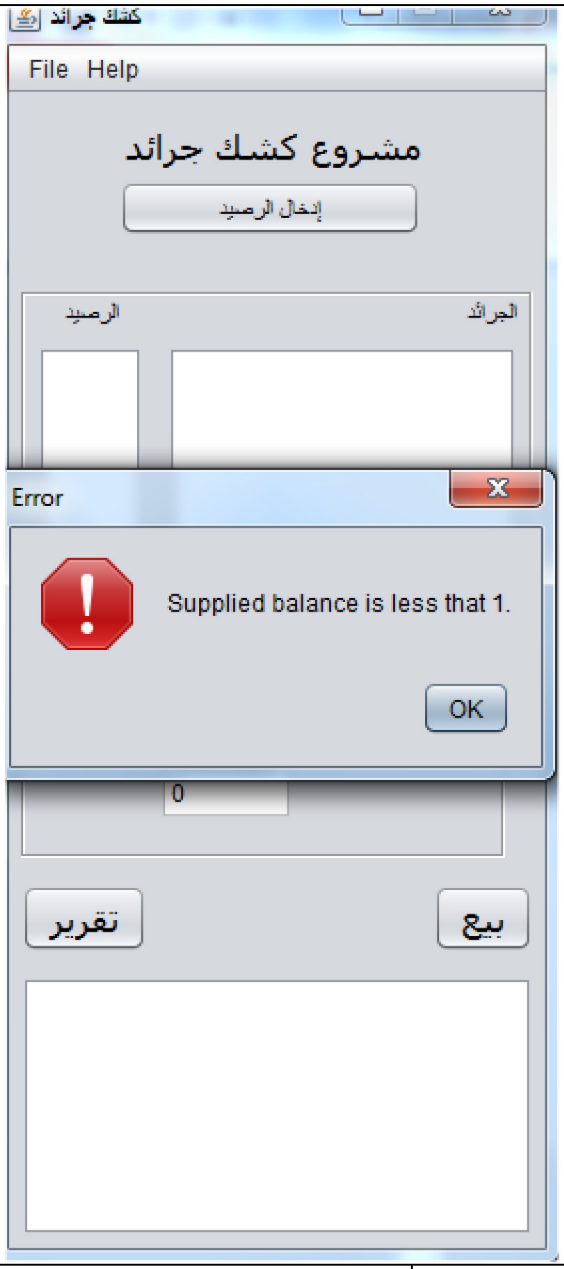
	CPointXY	void setX(int X) throws BoundriesException
		{
		if(X<0) throw new BoundriesException("Positive number required");
		this.X=X;
		}
		void setY(int Y) throws BoundriesException
		{
		if(Y<0) throw new BoundriesException("Positive number required");
		this.Y=Y;
		}

	Form	

	SetP1	try {
		if (Integer.parseInt(jTextField1.getText()) > jPanel4.preferredSize().width) {
		throw new BoundriesException("X is out of boundries, should be < " + jPanel4.preferredSize().width);
		}
		if (Integer.parseInt(jTextField2.getText()) > jPanel4.preferredSize().height) {
		throw new BoundriesException("X is out of boundries, should be < " + jPanel4.preferredSize().height);
		}
		t.setP1(Integer.parseInt(jTextField1.getText()), Integer.parseInt(jTextField2.getText()));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
		}
		jTextArea1.append("Point 1 : x = " +

		t.getP1().getX() + "\ty = " + t.getP1().getY() + "\n");

EXP11



BalanceException	public class BalanceException extends Exception{
	BalanceException()
	{
	super();
	}
	BalanceException(String msg)
	{
	super(msg);
	}

		@Override
		public String getMessage()
		{
		return "Supplied balance is less that 1.";
		}
		}

	MoneyException	public class MoneyException extends Exception{
		int type=0;
		MoneyException()
		{
		super();
		}
		MoneyException(String msg)
		{
		super(msg);
		}
		MoneyException(int type)
		{
		super();
		this.type=type;
		}
		@Override
		public String getMessage()
		{
		String msg;
		switch(type)
		{
		case 1:
		msg="supplied price is less that zero";
		break;
		case 2:
		msg="supplied prices results a profit is less that zero";
		break;
		default:
		msg="Undefined Error";

		break; } return msg; } }
	form	

	formWindowActivated	if (snpf.press.equalsIgnoreCase("apply")) {
		this.setEnabled(true);
		try {
		npb.addNewsPaper(snpf.name,
		Integer.parseInt(snpf.balance),
		Double.parseDouble(snpf.netPrice),
		Double.parseDouble(snpf.sellPrice));
		} catch (Exception e) {
		JOptionPane.showMessageDialog(this,
		e.getMessage(), "Error",
		JOptionPane.ERROR_MESSAGE);
		}
		DefaultListModel dlm1 = new
		DefaultListModel();
		DefaultListModel dlm2 = new
		DefaultListModel();
		for (int n = 0; n < npb.items; n++) {
		dlm1.addElement(npb.np[n].getName());
		}
		for (int n = 0; n < npb.items; n++) {
		
		dlm2.addElement(npb.np[n].getBalance());
		}
		
		jList1.setModel(dlm1);
		jList2.setModel(dlm2);
		} else if
		(snpf.press.equalsIgnoreCase("cancel")) {
		this.setEnabled(true);
		} else if (snpf.press.equalsIgnoreCase("1")) {

		snpf.setVisible(true);
		}
		snpf.press = "0";

EXP11	

