Activity: Static or Dynamic?

Use the code in this handout to follow along with the slides and answer the questions.

1. UML Diagram

In the space below, draw a UML diagram for `Sinkable`, `Pos`, `Boat`, `MotorBoat`, and `SailBoat`.

You can **leave the classes blank** (omit attributes and methods), but make sure to **draw the relationships**. Also indicate whether each type is an interface, abstract class, or concrete class.

2. Questions

Use this page to answer the poll questions (in the slides) and take notes.

Review 1: What kind of error would be produced? Why?

Review 2: What kind of error would be produced? Why?

Review 3: What kind of error would be produced? Why?

Review 4: What kind of error would be produced? Why?

Review 5: What would this code output?

Challenge 1: Which method would be called by this code?

Challenge 2: Which method would be called by this code?

Challenge 3: Which method would be called by this code?

Challenge 4: Which method would be called by this code?

Test 1: Which method would be called by this code?

Test 2: What would this code output?

3. Source Code

Sinkable.java

```java
public interface Sinkable {

    void sink();

}
```

Pos.java

```java
public class Pos {

    private int x;
    private int y;

    public Pos(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public String toString() {
        return "x: " + x + " y: " + x;
    }

}
```

Boat.java

```java
public abstract class Boat implements Sinkable {

    private static int count = 0;
    private String name;
    private Pos position;
    private boolean isAboveWater;

    public Boat(String bName) {
        this.name = bName;
        position = new Pos(0, 0);
        isAboveWater = true;
        count = count + 1;
    }

    public String getName() {
        return name;
    }

    public Pos getPos() {
        return position;
    }

    public int getCount() {
        return count;
    }

    public void setPos(Pos newPos) {
        if (isAboveWater) {
            position = newPos;
        }
    }

    public abstract Pos move();

    public void sink() {
        isAboveWater = false;
    }

    public String toString() {
        return String.format("Boat: %s %s Count: %d", name, position, count);
    }

}
```

MotorBoat.java

```java
public class MotorBoat extends Boat {

    private int speed;

    public MotorBoat(String mbName, int speed) {
        super(mbName);
        this.speed = speed;
    }

    public Pos move() {
        int newX = super.getPos().getX() + speed;
        int newY = super.getPos().getY() + speed;
        Pos newPos = new Pos(newX, newY);
        super.setPos(newPos);
        return super.getPos();
    }

}
```

SailBoat.java

```java
public class SailBoat extends Boat {

    public static final int WIND_SPEED = 10;

    private int sailNum;

    public SailBoat(String sName, int sailNum) {
        super(sName);
        this.sailNum = sailNum;
    }

    public Pos move() {
        int newX = super.getPos().getX() + WIND_SPEED;
        int newY = super.getPos().getY() + WIND_SPEED;
        Pos newPos = new Pos(newX, newY);
        super.setPos(newPos);
        return super.getPos();
    }

    public Pos sail() {
        Pos newPos = super.getPos();
        for (int i = 0; i < sailNum; i++) {
            newPos = move();
        }
        return newPos;
    }

    public String toString() {
        return String.format("Sailboat: %s %d", super.getName(), sailNum);
    }

}
```