October 7, 2005 Name ___________________

CS241 Exam 1

Fall 2005

1. (max = 10)	5. (max = 10)
2. (max = 10)	6. (max = 10)
3. (max = 10)	7. (max = 10)
4. (max = 10)	8. (max = 10)
Total Score ________(max=80)	

Final grade: ________/ 100

1) (10 pts) Define polymorphism.

2) (10 pts total)The value of *pi* can be found in the Math class where it is stored as a field called `PI` which is both `static` and `final`.

   a) (3 pts) What does `static` mean? Give as much detail as possible.

   b) (3 pts) What does `final` mean?

   c) (4 pts) Why does it make good programming sense to make `PI` both static and final.

3) (10 pts) What are the basic stages of the software development cycle. Which stage is most important and why?

4) (10 pts total) True/False and Multiple Choice:

   a) (6 pts) Subclasses have access to methods and fields in the superclass that have the following access (circle all that apply):

      private , protected , public

   b) (2 pts) Subclasses inherit <u>all</u> fields and methods from the superclass: True or False

   c) (2 pts) A stack follows the LIFO policy: True or False

5) (10 pts) Suppose you have an ArrayList object called nums:

```
ArrayList<Integer>  nums = new ArrayList<Integer>();
for (i=0;i<100;i++) nums.add((int) (100*Math.random());
```

Add code after the above for-loop that will remove all even numbers from `nums`.

6) (10 pts) One way to implement a queue is to use 2 stacks: `front` and `back`. . You add items by pushing onto `front` and you remove items by popping off of `back`. Complete the implementation below by completing the code for the `add`, `isEmpty`, and `remove` methods. Note, the challenge is to deal with the situation where `front` is empty, `back` is not, and we want to remove an item.

```
public class DoubleStackQueue<E> implements Queue<E> {

    java.util.Stack<E> front = new java.util.Stack<E>();
    java.util.Stack<E> back = new java.util.Stack<E>();

    public void add(E target) {




    }

    public boolean isEmpty() {




    }

    public E remove() {









    }
}
```

7) (10 pts total) Below is a programmer's attempt to roll two dice and order the results so that d1 contains the smaller roll. For example, if line 6 outputs

```
            die1 = 5    die2 = 3
```

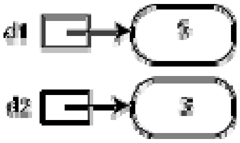
then line 8 should be

```
            die1 = 3    die2 = 5
```

```
line1:  public void test() {
line2:       Die d1 = new Die();
line3:       d1.roll();
line4:       Die d2 = new Die();
line5:       d2.roll();
line6:       System.out.println("die1 = "+d1+"   die2 = " + d2);
line7:       order(d1, d2);
line8:       System.out.println("die1 = "+d1+"   die2 = " + d2);
line0:  }

line10: public void order(Die a, Die b) {
line11:      Die c, d;
line12:      if (a.get() < b.get()){
line13:          c = a;
line14:          d = b;
line15:      } else {
line16:          c = b;
line17:          d = a;
line18:      }
line19:      a.set(c.get());
line20:      b.set(d.get());
line21: }
```

a) (4 pts) The picture at line 6 looks like:



To the above picture, add the references a, b, c, and d as they would appear just before line 19 is executed.

b) (4 pts) Redraw the above picture to show the state of d1, d2, a, b, c, and d just after line 20 is executed.

a) (2 pts) Does this code work properly? What is the output at line 8:

8) (10 pts) Given the Die class:

```
public class Die  {
  private int top=1;
  public int getTop() {
    return top;
  }
  public void roll() {
    top = ((int)(Math.random() * 6)) + 1;
  }
  public void setTop(int t) {
    top = t;
  }
  public String toString() {
    return "" + top;
  }
}
```

Below, write a new class called `PolyhedralDie` that *extends* the `Die` class and which can have any number of sides.