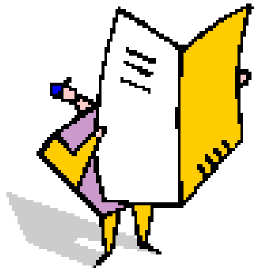
A+ Computer Science

AP REVIEW
2014 FR QUESTIONS

Free Response

-Read all 4 questions before writing anything

- -answer the easiest question 1st
- -most times question 1 is the easiest
- -see if part B calls part A and so on
- -many times part C consists of A and B calls
- -write something on every question
- -write legibly / use PENCIL!!!!!!!!!!
- -keep track of your time



Free Response

-When writing methods

  -use parameter types and names as provided

  -do not redefine the parameters listed

  -do not redefine the methods provided

  -return from all return methods

  -return correct data type from return methods

Free Response

-When writing a class or methods for a class

- -know which methods you have
- -know which instance variables you have
- -check for public/private on methods/variables
- -return from all return methods
- -return correct data type from return methods

Free Response

-When extending a class

  -know which methods the parent contains

  -have the original class where you can see it

  -make sure you have super calls

  -check for public/private on methods/variables

  -make super calls in sub class methods as needed

Free Response

-When extending abstract / implementing interface

-know which methods the parent contains

-have the original class where you can see it

-make sure you have super calls

-check for public/private on methods/variables

-make super calls in sub class methods as needed

-implement all abstract methods in sub class

Free Response Topics

ArrayList of References / Strings

– get,set,remove,add,size – levels of abstraction

GridWorld or Make a Class

– location, actor, bug, critter, ROCK, grid, super, abstract

Matrix / 2 D Array

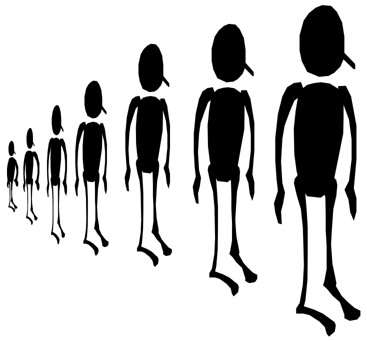
– nested loops, GridWorld (grid)

Make a Class / Interfaces / Abstract

– implement / extend – not seen this ? type in a few years

ArrayList

A typical ArrayList question involves putting something into an ArrayList and removing something from an ArrayList.



ArrayList

Arraylist is a class that houses an array.

An ArrayList can store any type.

All ArrayLists store the first reference at spot / index position 0.

ArrayList

```
int[] nums = new int[10];        //Java int array
```

	0	1	2	3	4	5	6	7	8	9
nums	0	0	0	0	0	0	0	0	0	0

An array is a group of items all of the same type which are accessed through a single identifier.

ArrayList

frequently used methods

Name	Use
add(item)	adds item to the end of the list
add(spot,item)	adds item at spot – shifts items up->
set(spot,item)	put item at spot z[spot]=item
get(spot)	returns the item at spot return z[spot]
size()	returns the # of items in the list
remove()	removes an item from the list
clear()	removes all items from the list

```
import  java.util.ArrayList;
```

ArrayList

```
List<String> ray;
ray = new ArrayList<String>();
ray.add("hello");
ray.add("whoot");
ray.add("contests");
out.println(ray.get(0).charAt(0));
out.println(ray.get(2).charAt(0));
```

OUTPUT

```
h
c
```

ray stores String references.

ArrayList

```
int spot=list.size()-1;
while(spot>=0)
{
   if(list.get(spot).equals("killIt"))
      list.remove(spot);
   spot--;
}
```

ArrayList

```
for(int spot=list.size()-1; i>=0; i--)
{
  if(list.get(spot).equals("killIt"))
    list.remove(spot);
}
```

ArrayList

```
int spot=0;
while(spot<list.size())
{

  if(list.get(spot).equals("killIt"))
    list.remove(spot);
  else
    spot++;

}
```

2014 Question 1 - part A

```
public  String scrambleWord( String word )
{
  String ret = "";
  for( int i = 0; i < word.length(); i++ )
  {
    if( i+1 != word.length()
          && word.substring(i,i+1).equals("A")
                  && !word.substring(i+1,i+2).equals("A"))
    {
      ret += word.substring(i+1,i+2) + word.substring(i,i+1);
      i++;       //prevents hitting the same "A" again
    }
    else
    {
      ret += word.substring(i,i+1);
    }
  }
  return ret;
}
```

You must know String!

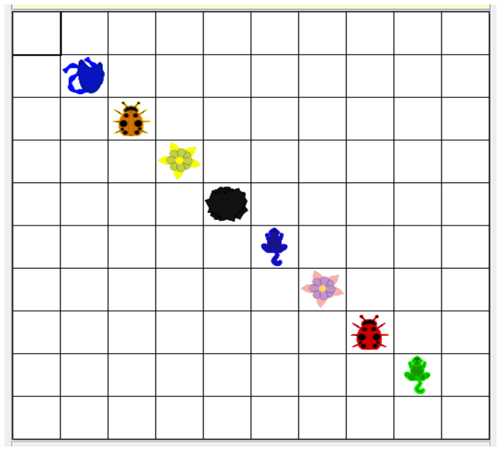
2014 Question 1 - part B

```
public void scrambleOrRemove( List<String> wordList )
{
   for( int i = wordList.size()-1; i >= 0; i--)
  {
     String cur = wordList.get( i );
     String ret = scrambleWord( cur );
     if( ret.equals( cur ) )
         wordList.remove( i );
     else
         wordList.set( i , ret );
   }
}
```

You must know ArrayList!

Matrices

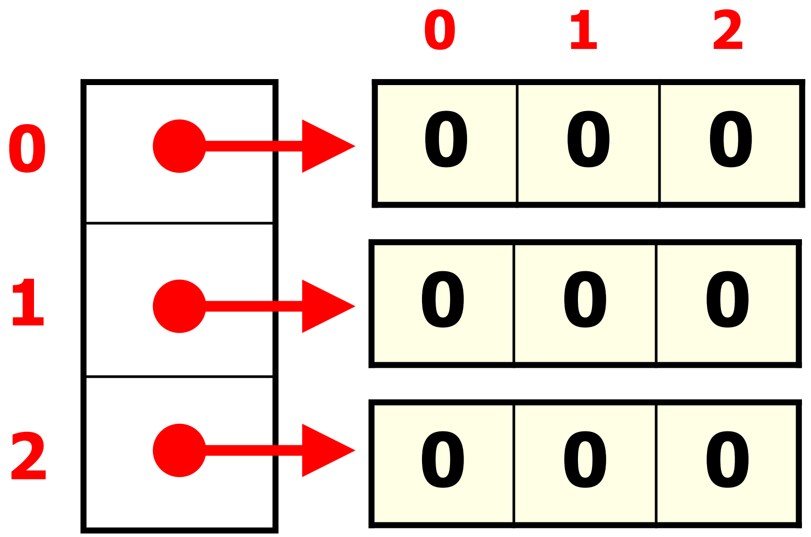
One question on the A test free response will require you to manipulate a 2-dimensional array or a GridWorld grid.



Matrices

A matrix is an array of arrays.

```
int[][] mat = new int[3][3];
```



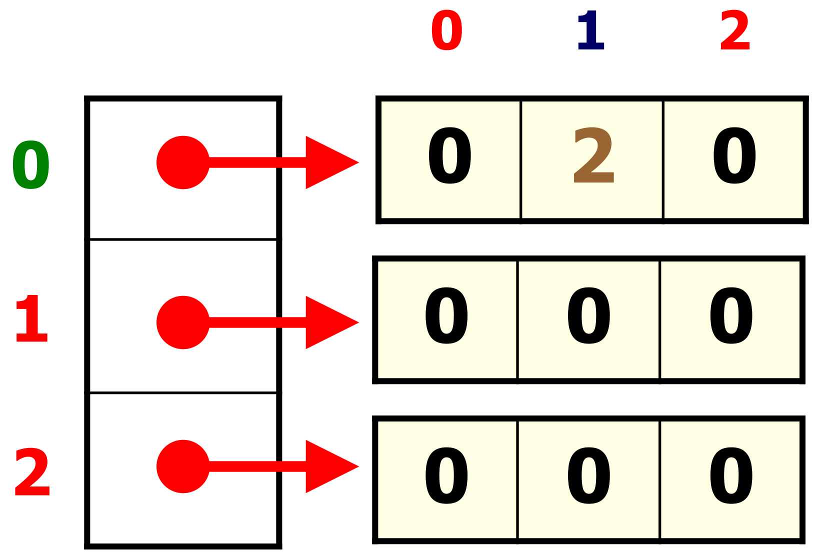
Matrices

A matrix is an array of arrays.

```
int[][] mat = new int[3][3];
mat[0][1]=2;
```

Which array?

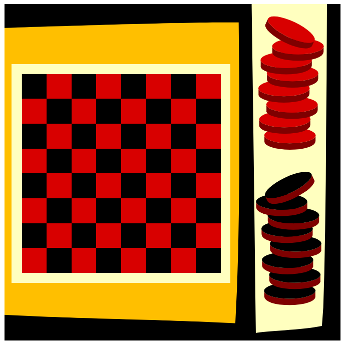
Which spot?



Matrices

$$
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 \\
\hline
0 & 0 & 0 & 0 & 5 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 7 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 3 & 0 & 0 & 0
\end{array}
$$

```
mat[2][2]=7;
mat[0][3]=5;
mat[4][1]=3
```



Matrices

```
for( int r = 0; r < mat.length; r++)
{
  for( int c = 0; c < mat[r].length; c++)
  {
       mat[r][c] = r*c;
  }
}
```

if mat was 3x3

0	0	0
0	1	2
0	2	4

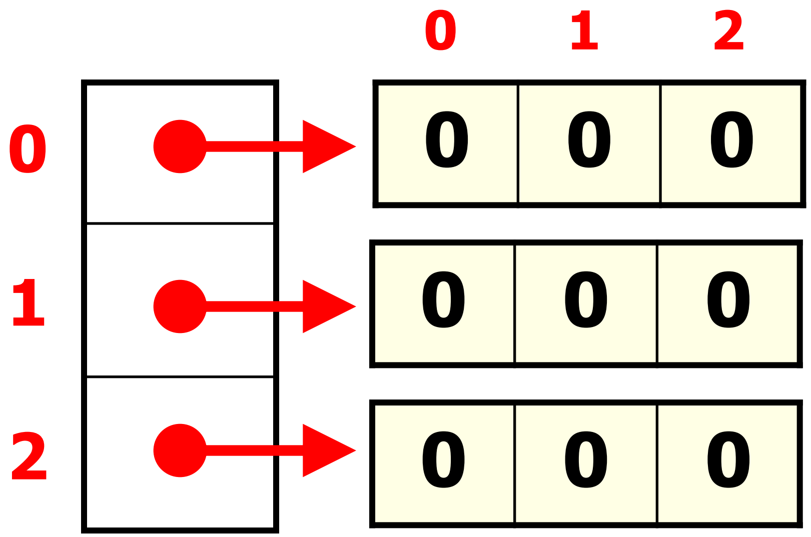
Matrices

A matrix is an array of arrays.

```
int[][] mat = new int[3][3];
```

# of arrays

size of each array



Matrices

```
int[][] mat = {{5,7},{5,3,4,6},{0,8,9}};

for( int[] row : mat )
{
   for( int num : row )
   {
      System.out.print( num + " ");
   }
   System.out.println();
}
```

OUTPUT

```
5 7
5 3 4 6
0 8 9
```

```
public SeatingChart( List<Student> studentList, int rows, int cols)
{
  seats = new Student[ rows ] [ cols ];
  int i = 0;
  for( int c = 0; c < seats[0].length; c++)
  {
    for( int r = 0; r < seats.length; r++)
    {
        if( i < studentList.size() )
           seats[r][c] = studentList.get( i++ );
    }
  }
}
```

This could be optimized, but it works perfectly and I assume many students are going to write something close to this.

2014 Question 3 part A – ver 1

```
public SeatingChart( List<Student> studentList, int rows, int cols)
{
   seats = new Student[ rows ] [ cols ];
   int i = 0;
   boolean stop = false;
   for( int c = 0; c < seats[0].length && !stop; c++)
   {
      for( int r = 0; r < seats.length; r++)
      {
            if( i < studentList.size() )
                 seats[r][c] = studentList.get( i++ );
            else        //added this in to make it more efficient
            {           //not required for AP CS A, but its fun to discuss
               stop = !stop;
               break;
            }
       }
   }
}
```

Here is the optimized version of ver 1. This not required, but has some fun stuff to discuss.

2014 Question 3 part A – ver 2

```
public SeatingChart( List<Student> studentList, int rows, int cols)
{
  seats = new Student[ rows ] [ cols ];

  for( int i = 0; i < studentList.size(); i++ )
  {
          //this algorithmic approach is common on lots
          //of matrix programming contest problems
     seats[ i % rows ][ i / rows] = studentList.get( i );
   }
}
```

This algorithm is really cool, but not one that most students would come up with on the exam. I teach this approach to my contest teams as there are often problems that involve storing strings in matrices at many contests.

2014 Question 3 part A – ver 3

2014 Question 3 - part B

```
public int removeAbsentStudents( int allowedAbsences )
{
  int count = 0;       //I stuck with column / row cuz I felt like it
  for( int c = 0; c < seats[0].length; c++)
  {
    for( int r = 0; r < seats.length; r++)
    {       //must check for null just like the Horse[] question from 2012
        if( seats[r][c] != null &&
                   seats[r][c].getAbsentCount()>allowedAbsences )
        {
          seats[r][c] = null;
          count ++;
        }
    }
  }
  return count;
}
```

Abstract / Interfaces

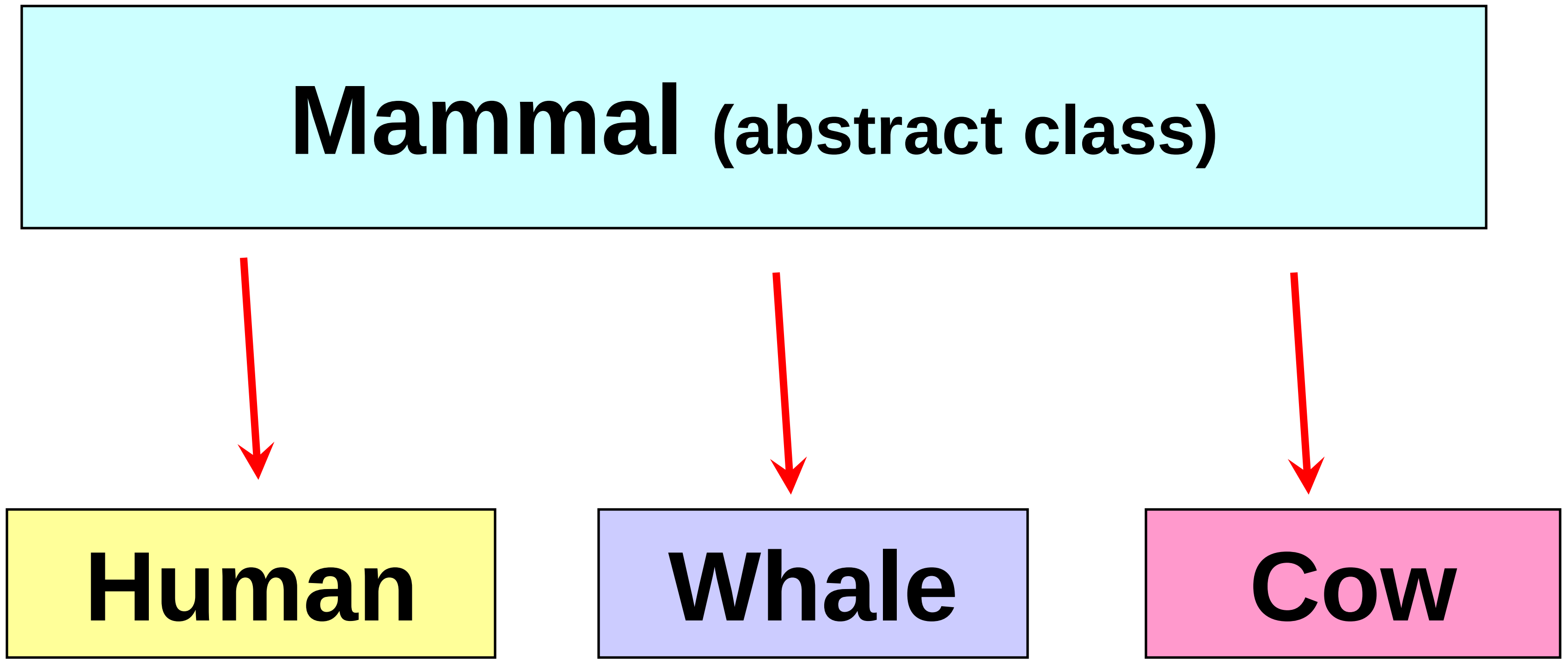
A typical Abstract/Interface question requires that a class be written that extends the abstract class or implements the interface and that all abstract method(s) be implemented.

Abstract / Interfaces

Abstract classes are used to define a class that will be used only to build new classes.

No objects will ever be instantiated from an abstract class.

Abstract / Interfaces



Abstract / Interfaces

Any sub class that extends a super abstract class must implement all methods defined as abstract in the super class.

Abstract / Interfaces

```
public abstract class APlus
{
  public APlus(int x)
    //constructor code not shown

  public abstract double goForIt();

  //other fields/methods not shown
}
```

Pet
Item

Abstract / Interfaces

```
public class PassAPTest extends APlus
{
  public PassAPTest(int x)
  {
    super(x);
  }

  public double goForIt()
  {
    double run=0.0;
    //write some code   -   run = x*y/z
    return run;
  }

  //other fields/methods not shown
}
```

```
public abstract class APlus
{
  public APlus(int x)
    //constructor code not shown

  public abstract double goForIt();

  //other fields/methods not shown
}
```

Abstract / Interfaces

```
public interface Exampleable
{
   int writeIt(Object o);
   int x = 123;
}
```

Methods are public abstract!
Variables are public static final!

Abstract / Interfaces

```
public interface Exampleable
{
  public abstract int writeIt(Object o);
  public static final int x = 123;
}
```

Methods are public abstract!
Variables are public static final!

Abstract / Interfaces

An interface is a list of abstract methods that must be implemented.

An interface may not contain any implemented methods.

Interfaces cannot have constructors!!!

Abstract / Interfaces

Interfaces are typically used when you know what you want an Object to do, but do not know how it will be done.

If only the behavior is known, use an interface.

Abstract / Interfaces

Abstract classes are typically used when you know what you want an Object to do and have a bit of an idea how it will be done.

If the behavior is known and some properties are known, use an abstract class.

```
public class Trio implements MenuItem
{
  private MenuItem one, two, three;     //I used MenuItem because that’s how I roll!

  public Trio( Sandwich f, Salad s, Drink t)   //Boo – constructor should take 3 MenuItems
  {
    one = f;
    two = s;
    three = t;
  }

  public String getName()
  {
    return one + "/" + two + "/" + three;
  }

  public double getPrice()
  {
     return Math.max( one.getPrice() + two.getPrice() ,
        Math.max( one.getPrice() + three.getPrice(), two.getPrice() + three.getPrice() ) );
  }

  public String toString()
  {
     return getName() +  " " + getPrice();
  }
}
```

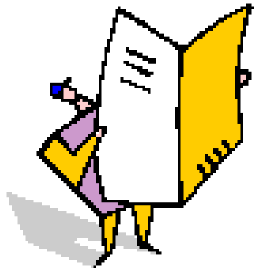
2014

Question 4

Free Response

-Read all 4 questions before writing anything

- -answer the easiest question 1st
- -most times question 1 is the easiest
- -see if part B calls part A and so on
- -many times part C consists of A and B calls
- -write something on every question
- -write legibly / use PENCIL!!!!!!!!!!
- -keep track of your time



Free Response

-When writing methods

- -use parameter types and names as provided
- -do not redefine the parameters listed
- -do not redefine the methods provided
- -return from all return methods
- -return correct data type from return methods

Free Response

- When writing a class or methods for a class
  - know which methods you have
  - know which instance variables you have
  - check for public/private on methods/variables
  - return from all return methods
  - return correct data type from return methods

Free Response

-When extending a class

- -know which methods the parent contains
- -have the original class where you can see it
- -make sure you have super calls
- -check for public/private on methods/variables
- -make super calls in sub class methods as needed

Free Response

-When extending abstract / implementing interface

 -know which methods the parent contains

 -have the original class where you can see it

 -make sure you have super calls

 -check for public/private on methods/variables

 -make super calls in sub class methods as needed

 -implement all abstract methods in sub class

Free Response Topics

ArrayList of References / Strings
– get,set,remove,add,size – levels of abstraction

GridWorld or Make a Class
– location, actor, bug, critter, ROCK, grid, super, abstract

Matrix / 2 D Array
– nested loops, GridWorld (grid)

Make a Class / Interfaces / Abstract
– implement / extend – not seen this ? type in a few years

A+ Computer Science

AP REVIEW
2014 FR QUESTIONS