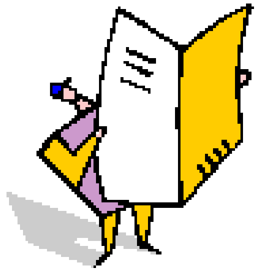
A+ Computer Science

AP REVIEW
2015 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

Array / Array or Arrays

– for and for each loops, nested loops, array of arrays concepts

Make a Class - Basic

– create a basic class with instance variables, constructors, and methods

ArrayList of References / Strings

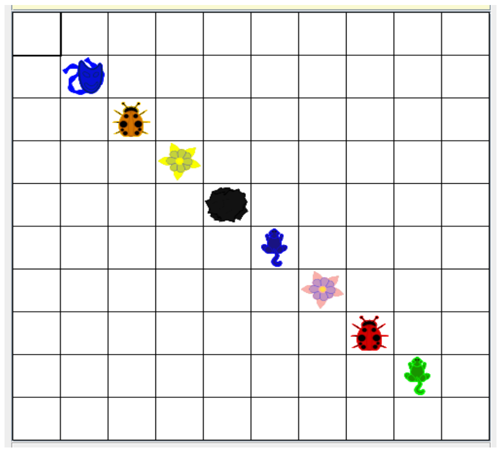
– get,set,remove,add,size – levels of abstraction

Make a Class - Interface / Abstract

– implement / extend – looks like this topic is back

Matrices

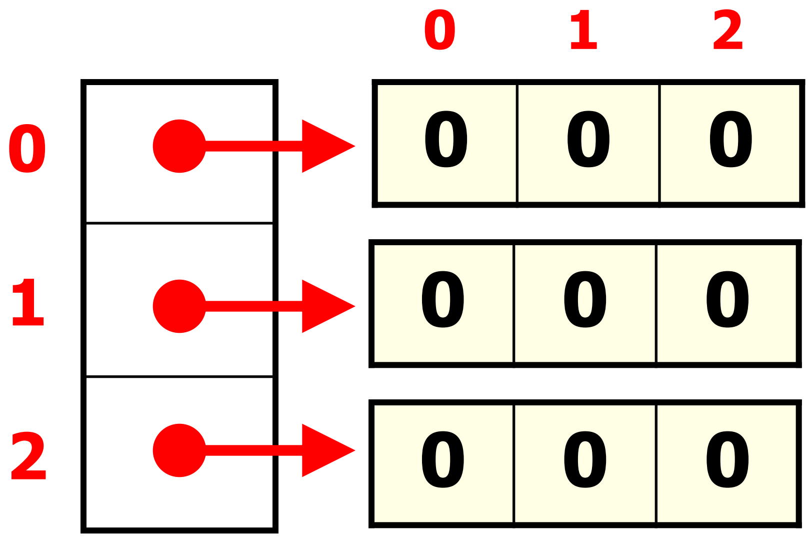
One question on the A test free response will require you to manipulate a 2-dimensional array or a GridWorld grid.



Matrices

A matrix is an array of arrays.

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



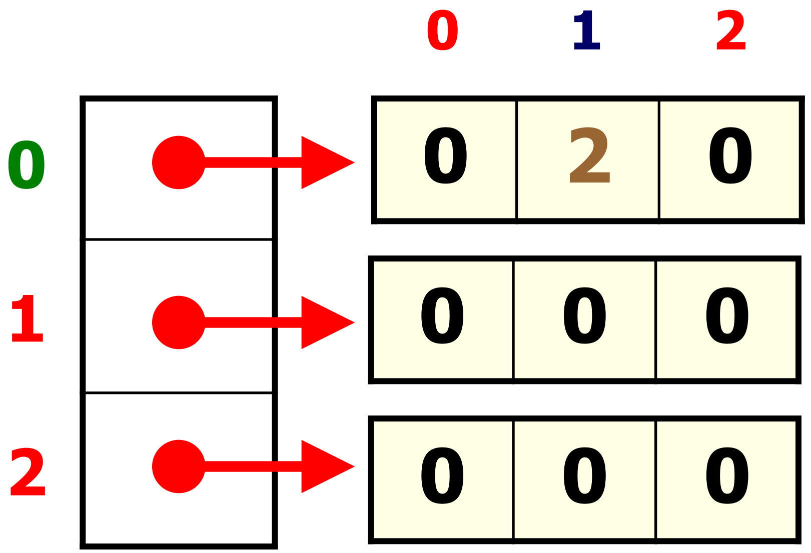
Matrices

A matrix is an array of arrays.

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

Which array?

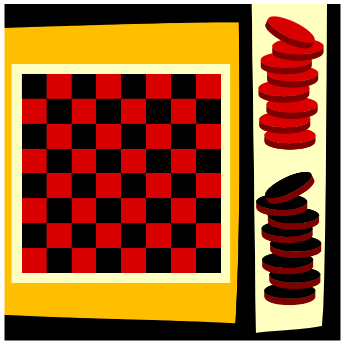
Which spot?



Matrices

	0	1	2	3	4
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

```
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;
  }
}
```

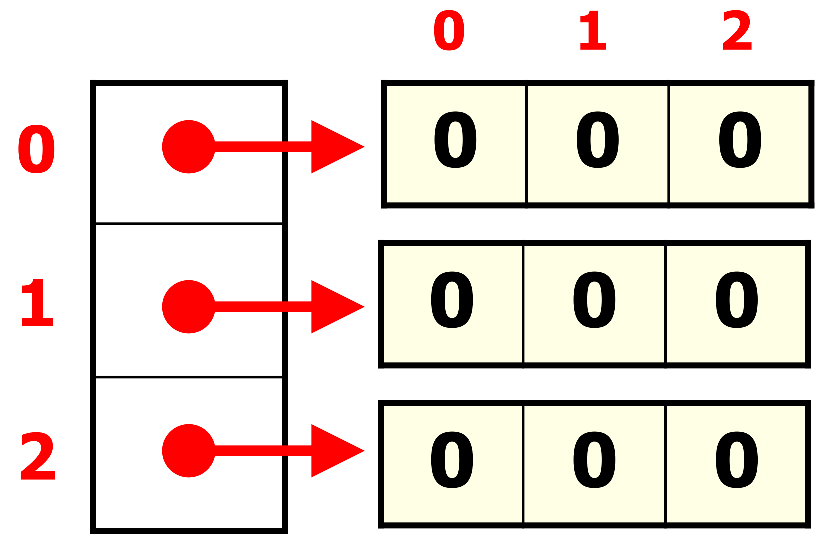
if mat was 3x3

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 static int arraySum( int[] arr )
{
    int sum = 0;
    for( int item : arr )
        sum = sum + item;
    return sum;
}
```

2015
Question 1
part A

```
public static int[] rowSums( int[][] arr2D )
{
  int[] ret = new int[arr2D.length];
  for( int i = 0; i < ret.length; i++ )
      ret[i] = arraySum(arr2D[i]);
  return ret;
}
```

2015 Question 1 part B

```
public static boolean isDiverse( int[][] arr2D )
{
  int[] totals = rowSums( arr2D );
  for( int i = 0; i < totals.length-1; i++ )
  {
     for( int j = i+1; j < totals.length; j++)
        if( totals[i] == totals[j] )
           return false;
  }
  return true;
}
```

2015 Question 1 part C – ver 1

This version is kinda lame, but I assume most students will do something like this.

```
public static boolean isDiverse( int[][] arr2D )
{
  int[] totals = rowSums( arr2D );
  ArrayList<Integer> used;
  used = new ArrayList<Integer>();
  for( int item : totals )
  {
     if( used.contains( item) )
        return false;
     else
        used.add( item );
   }
   return true;
}
```

2015 Question 1 part C – ver 2

This algorithm is more interesting and more of what I would expect from my students that participate in contests.

Make a Class

```
public class Triangle
{
   private int sideA;
   private int sideB;
   private int sideC;
```

Instance variables store the state information for an object.

Make a Class

```
public Triangle(int a, int b, int c)
{
   sideA=a;
   sideB=b;
   sideC=c;
}
```

Constructors are similar to methods. Constructors set the properties of an object to an initial state.

Make a Class

```
public void setSideA(int a )
{
    sideA=a;
}
```

Modifier methods are methods that change the properties of an object.

Make a Class

```
public int getSideA()
{
   return sideA;
}
```

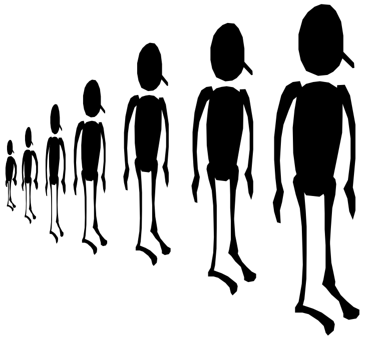
Accessor methods are methods that retrieve or grant access to the properties of an object, but do not make any changes.

2015 Question 2

```
public class HiddenWord
{
  private String word;

   public HiddenWord( String w )
   {
     word = w;
   }

   public String getHint( String guess )
   {
     String ret = "";
     for( int i = 0; i < word.length(); i++)
     {
           if( word.substring(i,i+1).equals(guess.substring(i,i+1)) )
             ret += word.substring( i, i+1);
           else if( word.indexOf( guess.substring(i,i+1))!=-1)
             ret += "+";
           else
             ret += "*";
     }
     return ret;
   }

}
```

ArrayList

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



ArrayList

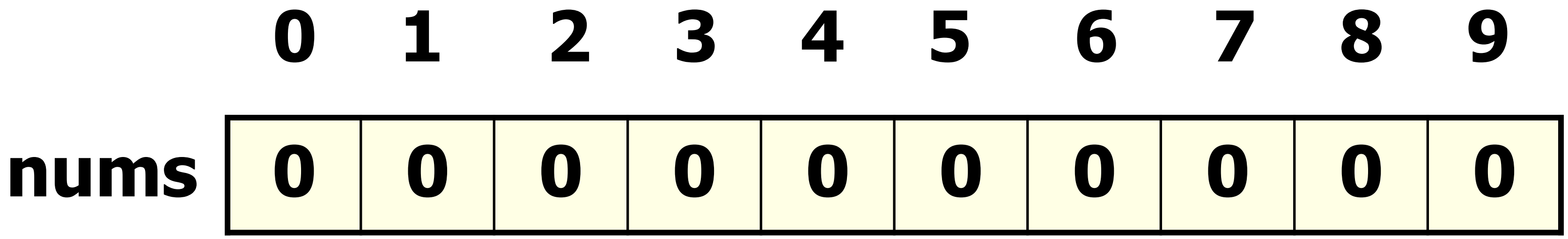
Arraylist is a class that houses an array.

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
```



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++;
}
```

2015 Question 3 part A

```
public int getValueAt( int row , int col )
{
  for( SparseArrayEntry item : entries )
  {
    if( item.getRow()==row && item.getCol()==col)
      return item.getValue();
  }
  return 0;
}
```

You must know ArrayList!

2015 Question 3 - part B

```
public void removeCol( int col )
{
   for( int i = entries.size()-1; i>=0; i--)
  {
     SparseArrayEntry e = entries.get(i);
     if( e.getCol()==col)
     {
       entries.remove(i);
      }
     else if ( e.getCol()>col)
     {
       SparseArrayEntry up;
       up = new SparseArrayEntry( e.getRow(), e.getCol()-1, e.getValue() );
       entries.set( i, up);
     }
  }
  numCols--;
}
```

You must know ArrayList!

Abstract / Interfaces

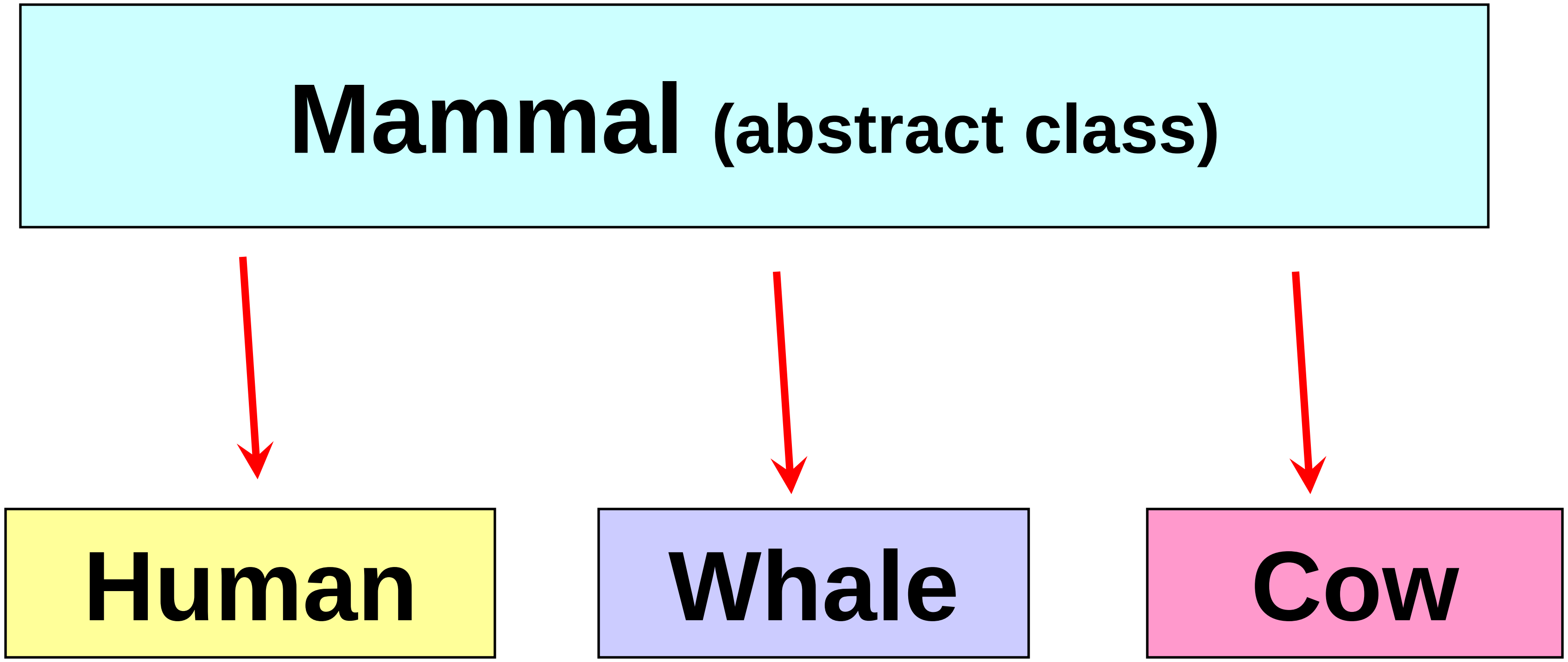
A typical Abstract/Interface question requires that a class be written that extends the abstract class or implements the interface and that all abstract method(s) be implemented.

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 interface NumberGroup
{
  boolean contains( int val );
}
```

2015
Question 4
Part A

2015 Question 4 Part B

```
public class Range implements NumberGroup
{
  private int low;
  private int high;

  public Range( int lo, int hi)
  {
    low = lo;
    high = hi;
  }

  public boolean contains( int val )
  {
    return val >= low && val <= high;
  }
}
```

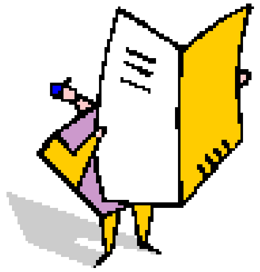
```
public class MultipleGroups implements NumberGroup
{
   private List<NumberGroup> groupList;

   //constructors , other methods, and such not shown

   public boolean contains( int val )
   {
      for( NumberGroup item : groupList )
      {
         if( item.contains( val) )
           return true;
       }
      return false;
   }
}
```

2015 Question 4 Part C

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

Array / Array or Arrays

– for and for each loops, nested loops, array of arrays concepts

Make a Class - Basic

– create a basic class with instance variables, constructors, and methods

ArrayList of References / Strings

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

Make a Class - Interface / Abstract

– implement / extend – looks like this topic is back

A+ Computer Science

AP REVIEW
2015 FR QUESTIONS