



AP[®] Computer Science A 2004 Sample Student Responses

The materials included in these files are intended for noncommercial use by AP teachers for course and exam preparation; permission for any other use must be sought from the Advanced Placement Program[®]. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. This permission does not apply to any third-party copyrights contained herein. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here.

The College Board is a not-for-profit membership association whose mission is to connect students to college success and opportunity. Founded in 1900, the association is composed of more than 4,500 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three million students and their parents, 23,000 high schools, and 3,500 colleges through major programs and services in college admissions, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT[®], the PSAT/NMSQT[®], and the Advanced Placement Program[®] (AP[®]). The College Board is committed to the principles of excellence and equity, and that commitment is embodied in all of its programs, services, activities, and concerns.

For further information, visit www.collegeboard.com

Copyright © 2004 College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Central, AP Vertical Teams, APCD, Pacesetter, Pre-AP, SAT, Student Search Service, and the acorn logo are registered trademarks of the College Entrance Examination Board. PSAT/NMSQT is a registered trademark of the College Entrance Examination Board and National Merit Scholarship Corporation. Educational Testing Service and ETS are registered trademarks of Educational Testing Service. Other products and services may be trademarks of their respective owners.

For the College Board's online home for AP professionals, visit AP Central at apcentral.collegeboard.com.

- (a) Given the class hierarchy shown above, write a complete class declaration for the class `Cat`, including implementations of its constructor and method(s). The `Cat` method `speak` returns "meow" when it is invoked.

```
public class Cat extends Pet
{
    public Cat (String name)
    {
        super(name);
    }
    public String speak()
    {
        return "meow";
    }
}
```

Part (b) begins on page 10.

GO ON TO THE NEXT PAGE.

- (b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String` *dog-sound* is returned by the `Dog` method `speak`, then the `LoudDog` method `speak` returns a `String` containing *dog-sound* repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

```
public class LoudDog extends Dog
{
    public LoudDog (String name)
    {
        super(name);
    }
    public String speak()
    {
        return super.speak() + " " + super.speak();
    }
}
```

GO ON TO THE NEXT PAGE.

(c) Consider the following partial declaration of class `Kennel`.

```

public class Kennel
{
    private List<Pet> petList; // all elements are references
                                // to Pet objects

    // postcondition: for each Pet in the kennel, its name followed
    //                 by the result of a call to its speak method
    //                 has been printed, one line per Pet
    public void allSpeak();
    { /* to be implemented in this part */ }

    // ... constructor and other methods not shown
}

```

Write the `Kennel` method `allSpeak`. For each `Pet` in the kennel, `allSpeak` prints a line with the name of the `Pet` followed by the result of a call to its `speak` method.

In writing `allSpeak`, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `allSpeak` below.

```

// postcondition: for each Pet in the kennel, its name followed
//                 by the result of a call to its speak method
//                 has been printed, one line per Pet
public void allSpeak()
{
    for (int i=0; i < petList.size(); i++)
    {
        Pet hold = (Pet)petList.get(i);
        System.out.println(hold.getName() + " " + hold.speak());
    }
}

```

GO ON TO THE NEXT PAGE.

- (a) Given the class hierarchy shown above, write a complete class declaration for the class `Cat`, including implementations of its constructor and method(s). The `Cat` method `speak` returns "meow" when it is invoked.

```
public class Cat extends Pet
{
    public Cat (String name)
        myName = name;
    public String speak()
        return "meow";
}
```

Part (b) begins on page 10.

GO ON TO THE NEXT PAGE.

- (b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String` *dog-sound* is returned by the `Dog` method `speak`, then the `LoudDog` method `speak` returns a `String` containing *dog-sound* repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

```
public class LoudDog extends Dog
{
    public LoudDog (String name)
        myName = name;
    public String speak()
        return dog-sound + dog-sound;
}
```

GO ON TO THE NEXT PAGE.

B₃

(c) Consider the following partial declaration of class `Kennel`.

```
public class Kennel
{
    [REDACTED] // all elements are references
                // to Pet objects

    // postcondition: for each Pet in the kennel, its name followed
    //                 by the result of a call to its speak method
    //                 has been printed, one line per Pet
    [REDACTED]
    { /* to be implemented in this part */ }

    // ... constructor and other methods not shown
}
```

Write the `Kennel` method `allSpeak`. For each `Pet` in the kennel, `allSpeak` prints a line with the name of the `Pet` followed by the result of a call to its `speak` method.

In writing `allSpeak`, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `allSpeak` below.

```
// postcondition: for each Pet in the kennel, its name followed
//                 by the result of a call to its speak method
//                 has been printed, one line per Pet
public void allSpeak()
```

```
{
    for (int i=0; i < petList.size(); i++)
        System.out.println(petList.get(i).getName() + " " +
            petList.get(i).speak());
}
```

GO ON TO THE NEXT PAGE.

A2 C1

- (a) Given the class hierarchy shown above, write a complete class declaration for the class `Cat`, including implementations of its constructor and method(s). The `Cat` method `speak` returns "meow" when it is invoked.

```
public class Cat extends Pet
{
    private String cName;
    public Cat (String name)
    {
        cName = name;
    }
    public String speak()
    {
        return "meow";
    }
}
```

Part (b) begins on page 10.

GO ON TO THE NEXT PAGE.

- (b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String` *dog-sound* is returned by the `Dog` method `speak`, then the `LoudDog` method `speak` returns a `String` containing *dog-sound* repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

```
public class LoudDog extends Dog
{
    private String dName;
    public Dog (String name)
    {
        dName = name;
    }
    public String speak()
    {
        String dogSound = "bark bark";
        return dogSound;
    }
}
```

GO ON TO THE NEXT PAGE.

C₃

(c) Consider the following partial declaration of class `Kennel`.

```
public class Kennel
{
    // all elements are references
    // to Pet objects

    // postcondition: for each Pet in the kennel, its name followed
    // by the result of a call to its speak method
    // has been printed, one line per Pet
    { /* to be implemented in this part */ }

    // ... constructor and other methods not shown
}
```

Write the `Kennel` method `allSpeak`. For each `Pet` in the kennel, `allSpeak` prints a line with the name of the `Pet` followed by the result of a call to its `speak` method.

In writing `allSpeak`, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method `allSpeak` below.

```
// postcondition: for each Pet in the kennel, its name followed
// by the result of a call to its speak method
// has been printed, one line per Pet
public void allSpeak()
```

```
public void allSpeak()
{
    for (int x = 0; x < petList.size(); x++)
    {
        System.out.println ( petList.get(x).getName() + " " +
            petList.get(x).speak() );
    }
}
```

GO ON TO THE NEXT PAGE.

- (a) Given the class hierarchy shown above, write a complete class declaration for the class `Cat`, including implementations of its constructor and method(s). The `Cat` method `speak` returns "meow" when it is invoked.

```
public class Cat extends Pet
{
    public Cat (string Name)
    {
        super.myName;
    }
    public String speak ()
    {
        return "meow";
    }
}
```

Part (b) begins on page 10.

GO ON TO THE NEXT PAGE.

- (b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String dog-sound` is returned by the `Dog` method `speak`, then the `LoudDog` method `speak` returns a `String` containing `dog-sound` repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

```
public class LoudDog extends Dog
{
    public LoudDog (String name)
    {
        super. myName;
    }
    public String speak()
    {
        super. speak;
        return speak + speak;
    }
}
```

GO ON TO THE NEXT PAGE.

D3

(c) Consider the following partial declaration of class Kennel.

```

public class Kennel
{
    [REDACTED] // all elements are references
                // to Pet objects
    // postcondition: for each Pet in the kennel, its name followed
    //                 by the result of a call to its speak method
    //                 has been printed, one line per Pet
    [REDACTED]
    { /* to be implemented in this part */ }

    // ... constructor and other methods not shown
}

```

XX
 Tiger Army
 Never Die!

Write the Kennel method allSpeak. For each Pet in the kennel, allSpeak prints a line with the name of the Pet followed by the result of a call to its speak method.

In writing allSpeak, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement functionality provided by these methods, rather than invoking these methods, will not receive full credit.

Complete method allSpeak below.

```

// postcondition: for each Pet in the kennel, its name followed
//                 by the result of a call to its speak method
//                 has been printed, one line per Pet
public void allSpeak()
{
    super.myName;
    super.speak;
    return myName + " " + speak + " " + myName + " " + speak + " " + myName + " " + speak;
}

```

GO ON TO THE NEXT PAGE.