

CS61B Summer 2006

Exall1 1

11 July 2006

Your name:

Your login: cs61b-

This exam is worth 25 points (out of a possible 230 for your course grade).

This exam is open book, open notes - you may use any references on paper that you wish, but no electronic devices (e.g. laptop computers) are permitted.

This booklet contains 11 numbered pages (including this page). Please make sure that you have all of the pages before you begin. You will have 80 minutes to complete the exam.

Please turn off your cell phones now.

1 What does Java output? (6 points)

There is a sheet attached to the back of the exam that you may tear off - the questions in this section will refer to code that is on that sheet.

What will the output be? Write on the line to the right of the code fragment. If it's an error, just write ERROR - you don't have to give the text of the error message. Suppose each code fragment is inside `main`. Each question is independent of the others (i.e. do not assume that the code from the previous questions has already run).

- ```
1. int i = 0 ;
 while i < 7 {
 if (i % 2 == 0) {
 continue ;
 }
 System.out.print(i) ;
 i = i + 1 ;
 }
```
- ```
2. MyClass a = new MyClass(2) ;
   MyClass b = new MyClass(4; )
   b = a ;
   a.setVar(17) ;
   System.out.println(b.getVar() ;
```
- ```
3. MyClass [] ma = new MyClass [4] ;
 ma [0].sayStuf f () ;
```
- ```
4. MyClass x = new MyClass(3);
   MyClass y = new MyClass(3);
   if (x == y) {
       System.out.println(“YES”);
   } else {
       System.out.println(“NO”);
   }
```

5.

```
try {
    throw new MyEx2;
}
catch (MyEx1 a) {
    System.out.println("catch1");
}
catch (MyEx2 b) {
    System.out.println("catch2");
}
```
6.

```
MyClass m = new ClassTwo() ;
m.saystuff ();
```

2 Multiple Choice (5 points)

Circle the letter for the correct answer. There is exactly one correct answer for each question.

- How do you permanently change the static type of a variable?
 - casting
 - assignment
 - overriding
 - not possible.
- A class declared as `final...`
 - can't extend another class
 - can't be extended
 - can't be instantiated
 - none of the above
- The value of a `static` field..
 - is the same for every instance of the class.
 - can never be changed.
 - must be set every time the class is instantiated
 - none of the above

4. Consider the following code:

```
public class Test {  
  
    public void method1() {  
        int x = 37;  
        method2 (x) ;  
    }  
  
    public void method2(int i) {  
        method3(i) ;  
        System. out .println(“here we are”) ;  
    }  
  
    public int method3(int j) {  
        return j + 1;  
    }  
  
    public static void main (String[] args) {  
        method1() ;  
    }  
}
```

We compile and run Test. When "here we are" prints, the variable x is ...

- (a) alive and in scope.
- (b) alive, but not in scope,
- (c) not alive, but in scope.
- (d) neither alive nor in scope.

5. Consider the following code:

```
public class K { }

public class F {
    public K k;
}

public class Test1 {
    public static void main (String[] args) {
        F f = new F();
        F[] fa = new F[2];
        Fa[0] = f ;
    }
}
```

When we run Test1, how many objects does main put on the heap?

- (a) 0
- (b) 1
- (c) 2
- (d) 3
- (e) more than 3

3 Inheritance Design (3 points)

Given 2 things, say how they would be related if we were writing a Java program to represent them. Write the letter of the appropriate description next to each question.

- a. A has an instance variable of type B
- b. A extends B
- c. B has an instance variable of type A
- d. B extends A

- 1. A: Kitchen B: House _____
- 2. A: Fruit B: Apple _____
- 3. A: Restaurant B: Menu _____

4 Identifying Errors (5 points)

Each of the following programs has an error in it and will not compile. For each program, describe in **one short phrase or sentence** what the error is.

```
1. public class SuperException extends Exception { }

    public class SubException extends SuperException { }

    public class Class1 {
        public void someMethod() throws SubException {
            System.out.println("this is Class1");
        }
    }

    public class Class2 extends Class1 {
        public void someMethod() throws SuperException {
            System.out.println("this is Class2");
        }
    }
```

```
2. public class Someclass {
    private int theInt;

    public static void main (String[] args) {
        System.out.println("The int is " + theInt);
    }
}
```

```
3. public class MyClass {
    private String message;

    public MyClass(String s) {
        message = s;
    }
}

public class Test {
    public static void main (String[] args) {
        MyClass c = new Myclass();
    }
}
```

```
4. public class MyClass {
    private String message;

    public MyClass(String s) {
        message = s;
    }
}

public class Test {
    public static void main (String[] args) {
        MyClass c = new MyClass("Hello");
        System.out.println(c.message);
    }
}
```

```
5. public class SomeException extends Exception { }

public class MyClass {

    public void doStuff(int i) throws SomeException {
        if (i < 10) {
            System.out.println("i less than 10");
        } else {
            System.out.println("i not less than 10");
            throw new SomeException();
        }
    }

    public void doMoreStuff() {
        System.out.println("hello");
        doStuff(3);
    }
}
```

5 Programming (6 points)

We're writing a program to represent students. Two kinds of students are highschoolers and college students. So far, we've written classes to represent students in general and high school students. High school students are like regular students, except that when they study, they complain about SATs instead of math.

```
public class Student {
    public float gpa;

    public void study() {
        System.out.println("Math is hard!");
    }

    public void procrastinate () {
        System.out.println("wasting time...");
    }
}

public class Highschooler extends Student {

    public void study() {
        System.out.println("I hate SATs!");
    }
}
```

1. College students are like regular students, except for two differences. First, they have to know which dorm they live in (represented by a String). Second, they study like other students, but procrastinate first. Write the CollegeStudent class here. (3 points)

```
public class CollegeStudent extends Student {

}

}
```

