

Object-Oriented Programming (in Java): Quiz 1.

Answer the questions on this sheet. As a final step, copy your answers to the attached answer sheet. You keep the original quiz paper and hand in the answer sheet.

1. What best describes the meaning of the following:

```
Actor aFrodo = new Hobbit();
```

- a) *aFrodo* is an *Actor* object.
 - b) The *Hobbit* class will be named *aFrodo*.
 - c) The object *aFrodo* is allocated on the *heap* and the variable named *Hobbit* is allocated on the *stack*.
 - d) *aFrodo* is a superclass object.
 - e) None of the above.
2. Which of the following is true about a *primitive* variable:
- a) The value stored in the variable is an actual value (as opposed to a reference to something).
 - b) It holds the raw machine-code address of a variable.
 - c) During program execution, accessing a primitive is slower than accessing a reference-based object.
 - d) Primitives are close to the machine-code level, thus have different byte sizes on different hardware platforms.
 - e) None of the above.
3. Which of the following is true about a *reference* variable:
- a) It holds the raw machine-code address of a variable, that gives access to objects stored somewhere else.
 - b) The space in memory consumed by the *reference* itself will vary depending on the number of fields in the *class*.
 - c) It can hold the location information of an object stored somewhere else.
 - d) The value stored in the variable is an actual value (such as the strength of an actor).
 - e) None of the above.
4. Which of the following statements about *constructors* is correct:
- a) A *constructor* has the same name as the class name.
 - b) A *constructor* is responsible for the initialization of an object's fields.
 - c) *Constructor* methods have no return type.
 - d) A class can have several *constructors*.
 - e) All of the above.
5. Which of the following is an application of the principle of *encapsulation*:
- a) the *Singleton* design pattern.
 - b) Fields (instance variables) are usually declared *private*.
 - c) Several methods have the same name, but have different *signatures*.
 - d) All classes are derived from the *superclass* called *Object*.
 - e) Objects created with *new* are allocated on the *heap*.

6. What is output by the following code:

```
public class Quiz {  
    public static void main(String[] args) {  
        String s1 = "Cricket";  
        System.out.print(change(s1));  
        System.out.print(s1);  
        s1 = change(s1);  
        System.out.print(s1);  
    } // end static void main()  
  
    public static String change(String s) {  
        s = "Changed";  
        return s;  
    }  
} // end class Quiz
```

Note: No
newlines are
generated by
this code.

The output is
guaranteed to
be on a single
line.

- a) ChangedCricketChanged
- b) CricketChangedCricket
- c) CricketCricketCricket
- d) ChangedCricketCricket
- e) nullCricketCricket

7. A *virtual method* absolutely requires which of the following to be in effect:

- a) The *subclass* overrides all methods in the ultimate *superclass* (called *Object*).
- b) The keyword *virtual* is applied to the method name.
- c) The *superclass* must be declared to be *abstract*.
- d) The method signatures must be identical in the *superclass* and *subclass*.
- e) All of the above.

8. What is output by the following code:

```
public class Quiz {  
    public static void main(String[] args) {  
        String s1 = "Jiminy";  
        String s2 = "Cricket";  
        s1 = s2 + s1;  
        s2 = s1;  
        s1 = "George";  
        System.out.println(s2);  
    } // end static void main()  
} // end class Quiz
```

- a) CricketJiminy
- b) George
- c) Cricket
- d) Jiminy
- e) None of the above.

Name: _____

CST8284: Quiz 1_a

Note: Fully fill your choice, for example: ☐ a ☐ b ☐ c ☒ d ☐ e

1. ☐ a ☐ b ☐ c ☐ d ☐ e

2. ☐ a ☐ b ☐ c ☐ d ☐ e

3. ☐ a ☐ b ☐ c ☐ d ☐ e

4. ☐ a ☐ b ☐ c ☐ d ☐ e

5. ☐ a ☐ b ☐ c ☐ d ☐ e

6. ☐ a ☐ b ☐ c ☐ d ☐ e

7. ☐ a ☐ b ☐ c ☐ d ☐ e

8. ☐ a ☐ b ☐ c ☐ d ☐ e