PART 01:

1. Create a new class called 'Item' with two protected instance variables (private variables), an integer variable called 'location', and a String variable called 'description'.

2. Add a constructor method for the Item class that takes an integer and a String as arguments (in that order).

3. The constructor should assign the value of these parameters to the corresponding instance variables.

4. Add getter and setter methods for the location and description variables.

5. Add another class called Monster and make the Monster class a sub-class of the Item class.

6. Add a constructor method to the Monster class that takes an integer and a String argument just like the Item class constructor.

7. Use these arguments to call the Item super class constructor from within the Monster class constructor so that the instance variables in the superclass are instantiated correctly.

Answer 1-7

```
public class Item {

  protected int location;

  protected String description;

  // Constructor

  public Item(int location, String description) {
        this.location = location;

        this.description = description;

    }

    // Getters and Setters

    protected int getLocation() {

        return location;

    }

    protected void setLocation(int location) {

        this.location = location;

    }

    protected String getDescription() {

        return description;

    }

    protected void setDescription(String description) {

        this.description = description;

    }

}
```

```
public class Monster extends Item {

    // Constructor calling the super class constructor

    public Monster(int location, String description) {

        super(location, description);

    }

}

public class Main {

    public static void main(String[] args) {

        // Create an Item object

        Item item = new Item(123, "Example Item");



        // Access Item instance variables using getters

        System.out.println("Item Location: " + item.getLocation());

        System.out.println("Item Description: " + item.getDescription());



        // Create a Monster object

        Monster monster = new Monster(456, "Scary Monster");
        // Access Monster instance variables using getters (inherited from Item)

        System.out.println("Monster Location: " + monster.getLocation());

        System.out.println("Monster Description: " + monster.getDescription());

    }

}
```

PART 02

Correct answer mark from blue color text and underline

1. Which of these keywords is used to refer to member of base class from a sub class?
   a) upper   [✓ b) super]   c) this   d) None of the mentioned

3. The modifier which specifies that the member can only be accessed in its own class is
   a) public   [✓ b) private]   c) protected   d) none

4. Which of these is a mechanism for naming and visibility control of a class and its content?
   a) Object   [✓ b) Packages]
   c) Interfaces   d) None of the Mentioned.

5. Which of the following is correct way of importing an entire package ‘pkg’?
   a) import pkg.   b) Import pkg.
   [✓ c) import pkg.*]   d) Import pkg.*

6. Which of these method of class String is used to extract a single character from a String object?
   a) CHARAT()   b) charat()
   [✓ c) charAt()]   d) CharAt()

7. Which of these method of class String is used to obtain length of String object?
   a) get()   b) Sizeof()
   c) lengthof()   [✓ d) length()]

PART 03: Fill in the blanks using appropriate term.

1. Real-world objects contain ___ and ___.
2. A software object's state is stored in ___.
3. A software object's behavior is exposed through ___.
4. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data ___.
5. A blueprint for a software object is called a ___.
6. Common behavior can be defined in a ___ and inherited into a ___ using the ___ keyword.
7. A collection of methods with no implementation is called an ___.
8. A namespace that organizes classes and interfaces by functionality is called a ___.
9. The term API stands for ___?

Answers

1.state and behavior.

2. in fields (instance variables).

3.methods.

4. encapsulation.

5. class.

6. superclass and inherited into a subclass using the extends keyword.

7. interface.

8. package.

9. Application Programming Interface**.**