For the questions on this review sheet, you should feel free to refer to the listing of methods in the Actor and World classes at the end of the review sheet and use them if needed to answer the questions. The same tables will be provided on the exam.

```
public class Foo {
    private int _num;
    private Bar _foo;

    public Foo() {
        _num = 0;
    }

    public void fooBar() {
        _foo = new Bar();
        moveForward(25);
    }
}
```

1. Use the class definition above to circle and identify the parts of code from the list given.

a) Constructor definition
b) Code that creates an object
c) Instance variable name
d) Instance variable declaration
e) Assignment statement
f) Method call
g) Method definition

2. What is the purpose of the constructor in code? When is a constructor called/executed?

The constructor is called every time you create a new object. The purpose of the constructor is to set up the initial state of the object.

For questions 3 - 7, fill assume that the method calls will go in the space indicated in the code sample given.

```
public class Forrest extends World {

  public Forrest() {

    //Code for Questions 3-7 would be written here

  }

}
```

3. Write the method call to add a Leaf to the world at location (45, 36).

```
addObject(new Leaf(), 45, 36);
```

4. Write the method call to add a Leaf to the world at a random location.

```
addObject(new Leaf(), Greenfoot.getRandomNumber(getWidth()),
Greenfoot.getRandomNumber(getHeight()) );
```

5. Write the method call to add a Leaf to the world at the lower right hand corner of the world.

```
addObject(new Leaf(), getWidth(), getHeight());
```

~~6. Write the code that adds 5 Leaf objects to the world at random locations.~~

7. Write the code that would remove all Bear objects from the world.

```
removeObjects(getObjects(Bear.class));
```

8. Write the code to create a Boy object.

```
new Boy();
```

9. Write the code to declare an instance variable of type Boy named _boy.

```
private Boy _boy;
```

10. Write the code that assigns the value 45 to a variable named temp.

```
temp = 45;
```

For questions 11 - 15, fill in the code in the class given.

```
public class Boy extends Actor {

   public void act() {
      move();
      checkForEdges();
      checkForCollisions();
   }

   public void move() {
      setLocation(getX()+1, getY()-2); //moves diagonally
      setLocation(getX()+1, getY()); //horizontal
      setLocation(getX(), getY()-2); //vertical
   }

   public void checkForEdges() {
      if(getX() == 0)
      {
      }
      else if(getX() >= getWorld().getWidth())
      {
      }

      if(getY() == 0)
      {
      }
      else if(getY() >= getWorld().getHeight())
      {
      }
   }

   public void checkForCollisions() {
      if(getOneIntersectingObject(Enemy.class) != null)
      {
          getWorld().removeObjects(getWorld().getObjects(Girl.class));

          Greenfoot.stop();
      }
   }
}
```

11. Fill in the move method so that the Boy moves in some direction. Think also about how the answer would change if you were required to move the Boy in a certain way. For example, move strictly horizontally across the screen, or strictly vertically on the screen.

12. Write the series of if-statements needed to check if the Boy is at the edges of the world.

13. Write the Java code required to check if the Boy has collided with an Enemy. If so, the scenario should stop.

14. Add to the code that before the scenario stops, all of the objects of type Girl should be removed from the world.

~~15. Instead of the code you wrote for questions 13 & 14, re-write the code for checkForCollisions to check for collisions with a Flower object. If the Boy collides with a flower, he should pick it up (remove it from the world). If the Boy has collected more than 2 flowers, another flower should be added to the world for every new flower he picks up. Otherwise, an Enemy should be inserted into the World at a random location.~~

Actor Method Summary

void	**act**() The act method is called by the greenfoot framework to give objects a chance to perform some action.
protected void	**addedToWorld**(World world) This method is called by the Greenfoot system when the object has been inserted into the world.
GreenfootImage	**getImage**() Returns the image used to represent this Actor.
protected java.util.List	**getIntersectingObjects**(java.lang.Class cls) Return all the objects that intersect this object.
protected java.util.List	**getNeighbours**(int distance, boolean diagonal, java.lang.Class cls) Return the neighbours to this object within a given distance.
protected java.util.List	**getObjectsAtOffset**(int dx, int dy, java.lang.Class cls) Return all objects that intersect the given location (relative to this object's location).
protected java.util.List	**getObjectsInRange**(int radius, java.lang.Class cls) Return all objects within range 'radius' around this object.
protected Actor	**getOneIntersectingObject**(java.lang.Class cls) Return an object that intersects this object.
protected Actor	**getOneObjectAtOffset**(int dx, int dy, java.lang.Class cls) Return one object that is located at the specified cell (relative to this objects location).
int	**getRotation**() Return the current rotation of the object.
World	**getWorld**() Return the world that this object lives in.
int	**getX**() Return the x-coordinate of the object's current location.
int	**getY**() Return the y-coordinate of the object's current location.
protected boolean	**intersects**(Actor other) Check whether this object intersects with another given object.
void	**setImage**(GreenfootImage image) Set the image for this object to the specified image.
void	**setImage**(java.lang.String filename) Set an image for this object from an image file.
void	**setLocation**(int x, int y) Assign a new location for this object.
void	**setRotation**(int rotation) Set the rotation of the object.

World Method Summary

void	**act**() Act method for world.
void	**addObject**(Actor object, int x, int y) Add an Actor to the world.
GreenfootImage	**getBackground**() Return the world's background image.
int	**getCellSize**() Return the size of a cell (in pixels).
java.awt.Color	**getColorAt**(int x, int y) Return the color at the centre of the cell.
int	**getHeight**() Return the height of the world (in number of cells).
java.util.List	**getObjects**(java.lang.Class cls) Get all the objects in the world, or all the objects of a particular class.
java.util.List	**getObjectsAt**(int x, int y, java.lang.Class cls) Return all objects at a given cell.
int	**getWidth**() Return the width of the world (in number of cells).
int	**numberOfObjects**() Get the number of actors currently in the world.
void	**removeObject**(Actor object) Remove an object from the world.
void	**removeObjects**(java.util.Collection objects) Remove a list of objects from the world.
void	**repaint**() Repaints the world.
void	**setActOrder**(java.lang.Class... classes) Set the act order of objects in the world.
void	**setBackground**(GreenfootImage image) Set a background image for the world.
void	**setBackground**(java.lang.String filename) Set a background image for the world from an image file.
void	**setPaintOrder**(java.lang.Class... classes) Set the paint order of objects in the world.
void	**started**() This method is called by the Greenfoot system when the execution has started.
void	**stopped**() This method is called by the Greenfoot system when the execution has stopped.