APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
Game class *Game class is where the main class that gets executed and contains your primary game logic*			
Implementation Required	`void setup()`	*Required Processing method that gets run once in your main class [from Processing's Structure class]*	Processing ref
Implementation Required	`void draw()`	*Required Processing method that automatically loops whatever is inside it. This MUST be called directly after `setup()`. (Anything drawn on the screen should be called from here) [from Processing's Structure class]*	Processing ref
Implementation Recommended	`void updateTitleBar()`	*Updates the Title Bar of the Game*	Stub provided
Implementation Recommended	`void updateScreen()`	*Updates what is drawn on the screen each frame. Can be called from draw()*	Stub provided
Implementation Recommended	`void populateSprites()`	*Populates the screen with desired Sprite images (ie. enemies or other characters)*	Stub provided
Implementation Recommended	`void moveSprites()`	*Moves all the Sprites on the screen each cycle*	Stub provided
Implementation Recommended	`boolean isGameOver()`	*Indicates when the game is over, returns true or false*	Stub provided
Implementation Recommended	`void endGame()`	*Used for game screen after the end of the game*	Stub provided
Implementation Optional	`void exampleAnimationSetup()`	*Sets up 5 horses to animate*	Stub provided
Implementation Optional	`void checkExampleAnimation()`	*Animates a set of horses to run across the screen only if the doAnimation variable is true*	Stub provided

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
Processing Helper Functions *These functions can be called directly as if written directly in the Game.pde file*			
Usage REQUIRED	`void size(int width, int height)`	*[Must be the first method called inside `setup()`] Defines the dimensions of the display window's width and height.*	Processing ref [Environment class]
Usage Optional	`void surface.setResizable(boolean b)`	*By default, Processing sketches can't be resized. When surface.setResizable(true) is used within a sketch, the window can be resized while it's running. [Structure class]*	Processing ref [Struture class]
Usage Optional	`void surface.setLocation(int x, int y)`	*Defines the position of the Processing sketch in relation to the upper-left corner of the computer screen [Structure class]*	Processing ref [Structure class]
Usage Recommended	`void surface.setTitle(String titleText)`	*Defines the title to appear at the top of the sketch window [Structure class]*	Processing ref [Structure class]
Usage Optional	`void fullScreen()`	*Maximizes the screen. Should only be used if not using a specific background image.*	Processing ref [Environment class]

Usage Optional	`void cursor(Pimage img)`
	`void cursor(Pimage img, int x, int y)`
	`void cursor(int kind)`
		Changes the cursor to an image or a special character. Parameter `x` and `y` will move the cursor to a specific active spot on the screen. `kind` can be ARROW, CROSS, HAND, MOVE, TEXT, or WAIT.	Processing ref [Environment class]

Usage Optional	`void noCursor()`	*Hides the cursor on the screen [Environment class]*	Processing ref [Environment class]
Usage Optional	`void noLoop()`	*Stops Processing from continuously executing the code within `draw()`.*	Processing ref [Structure class]
Usage Optional	`void exit()`	*Quits/stops/exits the program when called.*	Processing ref [Structure class]

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
	Mouse class *Processing Class to handle input from a computer mouse*		
Usage Recommended	`int mouseX`	*system variable that always contains the current horizontal coordinate of the mouse*	Processing ref
Usage Recommended	`int mouseY`	*system variable that always contains the current vertical coordinate of the mouse*	Processing ref
Usage Recommended	`int mouseButton`	*system variable that contains LEFT, RIGHT, or CENTER depending on which button is currently being pressed. Resets to 0 if no button is pressed*	Processing ref
Implementation Recommended	`void mousePressed()`	*Automatically runs ONCE whenever a mouse is pressed.*	Processing ref
Implementation Optional	`void mouseClicked()`	*Automatically runs AFTER a mouse is pressed and released.*	Processing ref
Implementation Optional	`void mouseReleased()`	*Automatically runs every time a mouse button is released.*	Processing ref
Implementation Optional	`void mouseWheel()`	*Automatically runs every time a mouse wheel moves.*	Processing ref
Implementation Optional	`void mouseMoved()`	*Automatically runs whenever the mouse moves and a button is NOT pressed.*	Processing ref
Implementation Optional	`void mouseDragged()`	*Automatically runs whenever the mouse moves AND a button IS pressed*	Processing ref

	Keyboard class *Processing Class to handle input from a computer keyboard*		
Usage Recommended	`char key`	*system variable that always contains the value of the most recent key pressed or released*	Processing ref
Usage Optional	`int keyCode`	*system variable that used to detect special keys, like UP, DOWN, LEFT, RIGHT arrow keys, ALT, CTRL, SHIFT*	Processing ref
Usage Optional	`boolean keyPressed`	*system boolean variable that returns true if any key is pressed and false if no keys are pressed*	Processing ref
Implementation Recommended	`void keyPressed()`	*Called once every time a key is pressed*	Processing ref
Implementation Optional	`void keyReleased()`	*Called once every time a key is released*	Processing ref
Implementation Optional	`void keyTyped()`	*Called once every time a key is pressed (IGNORING CTRL, SHIFT, and ALT keys)*	Processing ref

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
PImage class *Processing Class to handle images in your game*			
Usage Recommended	`PImage(String img)`	*Construct a new PImage object*	Processing ref
Usage Optional	`PImage(width, height, format, factor)`	*Construct a new PImage object*	Processing ref
Usage Required	`void image(PImage img, int x, int y)`	*draws an image to the display window*	Processing ref
Object Usage Optional	`void .resize(int width, int height)`	*Changes the size of an existing PImage to the specified `width` and `height`*	Processing ref
Object Usage Optional	`void .filter(int kind)`	*Applies a filter to the image, `kind` can be either `THRESHOLD, GRAY, OPAQUE, INVERT, POSTERIZE, BLUR, ERODE,` or `DILATE`*	Processing ref
Object Usage Optional	`boolean .save(String fileName)`	*Saves the image to a picture file fomat of .JPG or .PNG to the project's Sketch folder*	Processing ref
Usage Required	`PImage loadImage(String filePath)`	*Loads an image into a variable of type PImage. Four types of images (.gif, .jpg, .tga, .png) images may be loaded*	Processing ref
Usage Optional	`void imageMode(int mode)`	*Adjust which corner of the image is being referred to in the code. mode can be either CORNER, CORNERS, or CENTER*	Processing ref
Usage Optional	`PImage createImage(int w, int h, int format)`	*Creates a new image with width, `w`, and height, `h`, and format of either `RGB`, `ARGB`, or `ALPHA`*	Processing ref

PShape *Processing Class to handle shapes in your game*			
Usage Optional	`PShape(g, int kind, ... )`	*Construct a new PImage object*	Processing ref

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
SoundFile class Processing Class to handle sounds in your game. Must be imported with `import processing.sound.*;`			
Usage Optional	`SoundFile(this, String filepath)`	*Construct a new SoundFile Object. Can handle .wav, .aif, and .mp3 sound files*	Processing ref

Object Usage Recommended	`void .play()`
	`void .play(float rate, float amp)`
		Starts the playback of the soundfile. Only plays to the end of the audiosample once. If `cue()` or `pause()` were called previously, playback will resume from the cued position. Parameter `rate` refers to the speed of playback with `1.0` being normal speed. `amp` refers to the volume of the sound with `0.0` being silence & `1.0` being full volume.	Processing ref

Object Usage Optional	`void .cue(float time)`	*Cues the playhead to a fixed position in the soundfile. `time` refers to the seconds from the beginning to start the sound*	Processing ref
Object Usage Optional	`void .pause()`	*Stop the playback of the file, but cue it to the current position. The next call to `play()` will continue playing where it left off.*	Processing ref
Object Usage Optional	`void .loop()`	*Starts playback which will loop at the end of the soundfile.*	Processing ref
Object Usage Optional	`float .duration()`	*Returns the duration of the soundfile in seconds.*	Processing ref
Object Usage Optional	`void .isPlaying()`	*Checks whether this soundfile is currently playing*	Processing ref

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
Sprite class *Custom class to display a moveable sprite on the screen*			
Usage Required	`Sprite(String spriteImg, float x, float y)`	*Construct a Sprite object, with it's position as `x` and `y`, a path to the location of the image as `spriteImg`)*	Teacher provided
Object Usage Recommended	`void .show()`	*displays the Sprite on the screen*	Teacher provided
Object Usage Optional	`void .moveTo(float x, float y)`	*Moves Sprite image to a specific coordinate*	Teacher provided
Object Usage Recommended	`void .move(float x_change, float y_change)`	*Moves Sprite image incrementally from its current position*	Teacher provided
Object Usage Optional	`void .rotate(float degrees)`	*Rotates the image a certain number of degrees (90, 180, 270, 0)*	Teacher provided
Object Usage Optional	`float .getX()`	*Returns x coordinate of Sprite*	Teacher provided
Object Usage Optional	`float .getY()`	*Returns y coordinate of Sprite*	Teacher provided
Object Usage Optional	`PImage .getImg()`	*Returns the PImage of the Sprite*	Teacher provided
Object Usage Optional	`boolean .getIsAnimated()`	*Returns if the Sprite is an AnimatedSprite*	Teacher provided
Object Usage Optional	`void .setX(float x)`	*Sets the x position of the Sprite*	Teacher provided
Object Usage Optional	`void .setY(float y)`	*Sets the y postition of the Sprite*	Teacher provided
Object Usage Optional	`void .setImg(PImage img)`	*Sets the Sprite image*	Teacher provided
Object Usage Optional	`void .setIsAnimated(boolean a)`	*Sets if the Sprite is an AnimatedSprite*	Teacher provided

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
AnimatedSprite class *Custom class to display a Sprite that cycles through different poses on the screen*			
Usage Required	`AnimatedSprite(int x, int y, String png, String json)`	*Construct an AnimatedSprite object, which takes in the `x` & `y` coordinates of the top left corner of the Sprite, a `String png` for the filepath of a spritesheet with multiple images, and a `String json` that leads to a JSON file created from TexturePacker to tell where the different images are on the Spritesheet*	Teacher provided
Object Usage Optional	`void .show()`	*displays the AnimatedSprite on the screen*	Teacher provided
Object Usage Recommended	`void .animate(double animationSpeed)`	*Cycles through the images of the AnimatedSprite & shows on screen, based on the paramter `animationSpeed`, which should be a `double` between `0.0` and `1.0`*	Teacher provided
Object Usage Optional	`void .animateHorizontal(double hSpeed, double animationSpeed, boolean wraparound)`	*Animates & Moves an AnimatedSprite in a horizontal direction, using the `hSpeed` for movement, `animationSpeed` for how quickly to cycle through the images, and wraparound should be `true` if you want the image to re-appear on the opposite side if it goes off the edge or `false` if it disappears off the screen*	Teacher provided
Object Usage Optional	`void .animateVertical(double vSpeed, double animationSpeed, boolean wraparound)`	*Animates & Moves an AnimatedSprite in a horizontal direction, using the `vSpeed` for movement, `animationSpeed` for how quickly to cycle through the images, and wraparound should be `true` if you want the image to re-appear on the opposite side if it goes off the edge or `false` if it disappears off the screen*	Teacher provided

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
Grid Class — *Custom class to overlay a 2D Grid stucture over the screen*			
Usage Recommended	`Grid(int rows, int cols)`	*Grid constructor that will create a Grid with the specified number of `rows` and `cols`*	Teacher provided
Usage Optional	`Grid()`	*Grid constructor that will create a 3x3 Grid*	Teacher provided
Object Usage Optional	`void .setMark(String mark, GridLocation loc)`	*Assigns a `String mark` to a location in the `Grid`. This `mark` is not necessarily visible, but can help in tracking what you want recorded at each `GridLocation`.*	Teacher provided
Object Usage Optional	`boolean .setNewMark(String mark, GridLocation loc)`	*Assigns a `String mark` to a location in the `Grid`. This `mark` is not necessarily visible, but can help in tracking what you want recorded at each `GridLocation`. Returns `true` if mark is correctly set when there was not a previous mark or `false` if not*	Teacher provided
Object Usage Optional	`void .printGrid()`	*Prints out marks in the Grid to the console*	Teacher provided
Object Usage Optional	`GridLocation .getGridLocation()`	*Returns the `GridLocation` of where the mouse is currently hovering over*	Teacher provided
Object Usage Optional	`int .getX(GridLocation loc)`	*Accessor method that provide the x-pixel value given a `GridLocation loc`*	Teacher provided
Object Usage Optional	`int .getY(GridLoction loc)`	*Accessor method that provide the y-pixel value given a `GridLocation loc`*	Teacher provided
Object Usage Optional	`int .getRows()`	*Accessor method that returns the number of rows in the `Grid`*	Teacher provided
Object Usage Optional	`int .getCols()`	*Accessor method that returns the number of cols in the `Grid`*	Teacher provided
Object Usage Recommended	`GridTile .getTile(Gridlocation loc)`	*Returns the `GridTile` object stored at a specified `GridLocation`*	Teacher provided
Object Usage Optional	`GridTile .getTile(int r, int c)`	*Returns the GridTile object stored at a specified row `r` and column `c`*	Teacher provided

APCSA Java Game Design through Processing Ref Sheet

Accessible methods from the Processing classes provided in the Starter Code

Additional Processing commands can be found in the official Processing Reference site.

Usage	Class Constructors and Methods	Explanation	Reference
GridLocation Class — *Custom class to store information in locations in the Grid*			
Usage Recommended	`GridLocation(int row, int col)`	*GridLocation constructor, given row and column parameters*	Teacher provided
Object Usage Recommended	`int .getR()`	*Accessor method to get row value of GridLocation*	Teacher provided
Object Usage Recommended	`int .getC()`	*Accessor method to get column value of GridLocation*	Teacher provided

GridTile Class — *Customizable class to store a PImage and String in GridTiles on the Grid*			
Usage Recommended	`GridTile()`	*Default GridTile constructor which puts an " " mark in the `GridTile`*	Teacher provided
Usage Optional	`GridTile(String mark)`	*`GridTile` constructor which adds the specified `String mark`*	Teacher provided
Object Usage Recommended	`String .getMark()`	*Gets the mark in the GridTile*	Teacher provided
Object Usage Optional	`void .setMark(String mark)`	*Automatically changes the mark*	Teacher provided
Object Usage Recommended	`boolean .setNewMark(String mark)`	*Sets a new mark in the GridTile if it does not already have a mark, returns true or false if successful*	Teacher provided
Object Usage Optional	`void .setImage(PImage pi)`	*Sets an new PImage in the GridTIle*	Teacher provided
Object Usage Optional	`PImage .getImage()`	*Returns the PImage stored in the GridTile*	Teacher provided